SHADOW Version 1.8 Installation Manual

The SHADOW Team

<shadow@nswc.navy.mil>

25 April 2003

Abstract:

SHADOW is an Intrusion Detection System developed on inexpensive hardware and free software. This document details how to build, install and configure the system.

Table of Contents

1.0 Introduction 4
2.0 History 4
3.0 Open Source Software 5
4.0 Architectural Overview 5
4.1 Hardware 5
4.2 Software 7
4.2.1 Sensor 7
4.2.1.1 sensor_init.sh 7
4.2.1.2 sensor_driver.pl 7
4.2.1.3 stop_logger.pl 7
4.2.1.4 start_logger.pl 7
4.2.1.5 std.ph 7
4.2.2 Analyzer 7
4.2.2.1 fetchem.pl 8
4.2.2.2 cleanup.pl 8
4.2.2.3 condense.pl 8
4.2.2.4 obfuscate.pl 8
4.2.2.5 find_scan.pl 8
4.2.2.6 comment_strip 8
4.2.2.7 Statistics Scripts 8
4.2.2.7.1 statistics.ph 9
4.2.2.7.2 do_daily_stats.pl 9
4.2.2.7.3 print_stats.pl 9
4.2.2.8 The CGI Scripts 9
4.2.2.8.1 tools.cgi 9
4.2.2.8.2 search.cgi 9
4.2.2.8.3 whois.cgi 10
4.2.2.8.4 lookup.cgi 10
4.2.2.8.5 compose_IR.cgi 10
4.2.2.8.6 kill_group.cgi 10
4.2.2.8.7 nmap.cgi 10
4.2.2.9 JavaScript 10
4.2.2.9.1 openwin.js 10
5.0 Installation 12
5.1 How to Build a SHADOW Sensor. 12
5.1.1 Sensor Hardware Requirements. 12
5.1.2 Install and Patch the Sensor Operating System. 12
5.1.3 Build a Custom Sensor Kernel. 14
5.1.4 Create the SHADOW User Account on the Sensor. 15
5.1.5 Unpack the SHADOW-1.8 Software. 15
5.1.6 Install *Tcpdump* on the Sensor. 15
5.1.7 Install OpenSSH on the Sensor. 16
5.1.8 Configure SHADOW Sensor Software. 16
5.1.9 Configure Crontab and System Start-up Files for the Sensor. 17
5.1.10 Protect the Sensor. 18

5.1.11 Situate the Sensor. 19
5.2 How to Build a SHADOW Analyzer. 20
5.2.1 Analyzer Hardware Requirements. 20
5.2.2 Install and Patch the Analyzer Operating System. 21
5.2.3 Build a Custom Analyzer Kernel. 21
5.2.4 Create a SHADOW User Account on the Analyzer. 22
5.2.5 Unpack the SHADOW-1.8 Software 22
5.2.6 Install the Accessories. 23
5.2.6.1 Install and Configure Apache. 23
5.2.6.2 Install the SHADOW Home Web Page. 25
5.2.6.3 Install *Tcpdump* on the Analyzer. 25
5.2.6.4 Install and Configure OpenSSH. 26
5.2.6.5 Test the Analyzer to Sensor Connection 28
5.2.6.6 Install and Configure NMAP. 29
5.2.6.7 Configure SUDO. 29
5.2.7 Configure SHADOW Analyzer Software. 30
5.2.7.1 Configure Global Analyzer Parameters. 30
5.2.7.2 Configure the Individual Sensor (Site) Configuration Files. 33
5.2.7.3 Create the Required Directories. 36
5.2.7.4 Configure the *Tcpdump* Filters. 37
5.2.7.5 Configure the CGI Scripts. 37
5.2.8 Test the SHADOW Scripts. 38
5.2.9 Configure Crontab on the Analyzer. 39
5.2.10 Protect the SHADOW Analyzer. 40
5.3 Put the SHADOW System into Production 41
Appendix A: SHADOW Sensor Kernel Configuration 42
Appendix B: SHADOW Analyzer Kernel Configuration 47
Appendix C: Protecting the SHADOW System 52
Bibliography 57

1.0 Introduction

SHADOW is an Intrusion Detection system developed on inexpensive PC hardware running Open Source, public domain, or freely available software. A SHADOW system consists of at least two pieces: a sensor located at a point near an organization's firewall, and an analyzer inside the firewall. SHADOW performs traffic analysis; the sensor collects packet headers from all IP packets that it sees; the analyzer examines the collected data and displays user defined “interesting” events on a web page. SHADOW uses the *tcpdump* and *libpcap* software packages developed at the Lawrence Berkeley Laboratory to collect IP packets from the network and to filter them according to user defined criteria. Using Perl scripts developed at the Naval Surface Warfare Center Dahlgren Division, the results are displayed via an Apache web server. The key to effective use of SHADOW is intelligent definition of the *tcpdump* filters based on the network environment and educated recognition of known and potential exploits from traffic patterns.

This document outlines the steps for installing a SHADOW system. Given the diverse environments into which a SHADOW system may be installed, these steps are general and leave a lot of work for the installer. The installer must be intimately familiar with the architecture of his network, the workings of his Linux system and the process of installing open source software. Steps not fully detailed here will be showstoppers. For example, a system lacking a C compiler or Perl interpreter will not produce a working SHADOW system.

This document concentrates exclusively on installation for a Red Hat Linux system. That is the operating system on which SHADOW is developed and maintained. It is known to work on other systems, but, the procedures outlined here have been developed for and tested only on Red Hat Linux. Installation on other operating systems may require modification of these procedures.

2.0 History

Before Intrusion Detection became popular and hackers made the evening news, analysts at research and development sites were interested in examining the traffic traversing their networks. The Naval Surface Warfare Center/Dahlgren Division is one such site. But since computer security had not yet made it to the mainstream of consciousness, little money was available for buying or building custom devices to examine network traffic patterns. So, Stephen Northcutt, at NSWC, began building network sniffers from recycled Sun workstations and public domain software. He used a software package from Texas A & M University called “*netlog*,” which consists of three programs, *icmplogger*, *tcplogger*, and *udplogger* to collect appropriate IP packets. Auxiliary programs read the collected data and display it in human readable format. Stephen wrote a set of Perl scripts to parse the data and create web pages on which were written “events of interest” that his scripts found in the data.

At that point, SHADOW was born. It had not yet been named, and it had a lot of maturing to do. Using old Sun boxes proved to be limiting. They were functional but obsolete in terms of computing power and software support. As the network traffic increased, and the availability of Suns decreased, alternatives

were sought. NSWC discovered that performance could be improved by using newer PCs and Linux instead of recycled Suns. Unfortunately, the *netlog* package was not available for Linux on the Intel x86 hardware. A more capable package called *tcpdump* was available from Lawrence Berkeley Laboratory. The switch to *tcpdump* was made, and the project was named SHADOW. SHADOW was first built on Red Hat Linux Version 5.0, and it has been migrated to every subsequent version.

3.0 Open Source Software

In the days of large scientific computer mainframes like Control Data 6600s and Crays, many government sites did not depend on computer vendors to supply and support operating system software. In fact, when the first Cray machines were built, Department of Energy sites had a time-sharing operating system ready to run on the hardware before the vendor did. There were certainly advantages to this approach. Operation of the site computers was not dependent on the vendor's priorities or abilities. Security, which was a high priority of the government organization but not necessarily of the vendor, could be implemented from the start. If a problem occurred with the system software, the source code and usually its developers were available to devise a solution. With availability of the source code, (and someone technically astute enough to find and fix it), any software problem can be solved; without it, the buyer is at the mercy of a vendor.

From a security perspective, the only way to insure that a piece of software does what it's supposed to and only what it's supposed to do is to examine and understand the source. Commercial software companies will gladly sell their code in binary form and claim that it functions as advertised. But read the license. It says that the software may not do what it is supposed to do; it may have unpredictable, or even damaging, side effects on your hardware or other software; but the company is not responsible. Once you accept that license, the risk is yours alone. That says a lot about what such companies feel about quality and responsibility. Caveat emptor.

4.0 Architectural Overview

The SHADOW system consists of two functional pieces: sensors and analyzers. The sensors sit at network boundaries, where a local network is connected to an external one. They are positioned such that they can collect data about packets entering or leaving the local network. The data is saved in files that accumulate on the sensor's disks. Analyzers sit inside the local network, hopefully protected by a firewall, to periodically retrieve and remove the data files from the sensors. The analyzers examine the data to provide views of it from different perspectives, such as potential intrusion events and statistical traffic summaries.

4.1 Hardware

Figure 1 illustrates a possible network infrastructure into which a SHADOW system has been inserted.

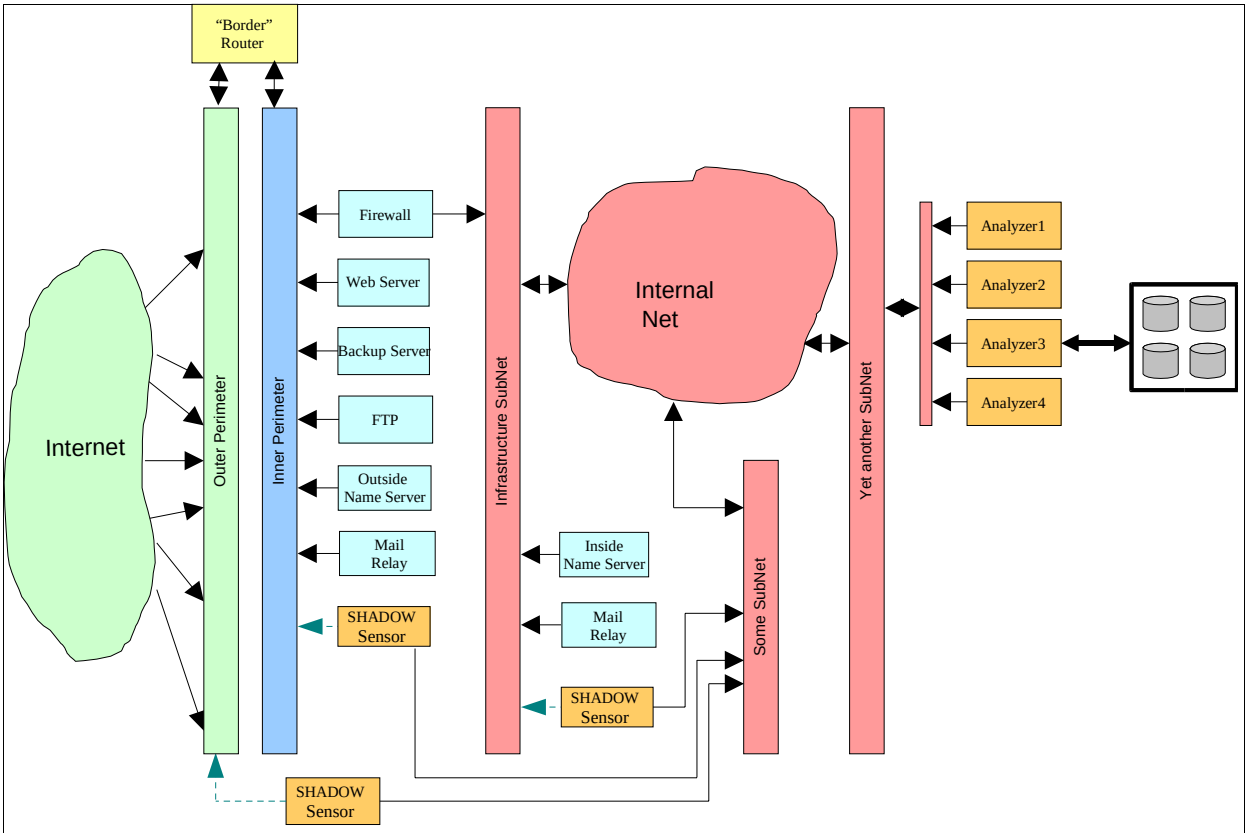

Figure 1 – Possible network infrastructure

On the left of the figure is the external Internet. The internal net is connected to the external Internet through a firewall, connected to an "Inner Perimeter" network that is connected via a "Border Router" to an "Outer Perimeter" network. Located in what's called a "DMZ" outside the firewall are other externally accessible servers, for mail, web, and name services, etc. Any traffic originating externally must pass through the firewall and its screening rules to reach the internal network.

This network has three SHADOW sensors installed, as shown in the bottom center of the figure. From the left of the drawing, one sensor is connected to the outer perimeter network to collect all externally originating packets as they first traverse into the media of the internal network. The second sensor collects packets on the inner perimeter network once they have passed through the border router and the screening defined by its filters. The third sensor collects packets on a sub-network inside the firewall, after they have been screened by the rules enforced by the firewall. All three sensors have dual network interfaces. One interface is "invisible," in that it has no IP address and is merely collecting packets from the perimeter networks. The other interface is connected inside the firewall so that the SHADOW analyzers can fetch the sensors' data files without interfering with the sensors' packet gathering or placing unnecessary loads on the firewall.

Well inside the firewall is a SHADOW analyzer, or possibly a cluster of them, as shown at the right of the figure. Members of the cluster may share a RAID system to provide storage for the data files. SHADOW analysis work may be shared among the systems of the cluster to provide improved performance through load balancing and redundancy.

4.2 Software

The SHADOW software system comprises a collection of Perl scripts. The scripts on the sensors collect the IP packet data using *tcpdump,* and save it in hourly chunks. Scripts on the analyzers fetch the data from the sensors, run it through *tcpdump* using filters to create web pages with "events of interest" and statistical summaries for analysis. There are also scripts on the analyzers to provide tools to assist in more detailed examination of the data. The next two sections will list each of the script names in the SHADOW package and provide a brief description of its functionality.

4.2.1 Sensor

The following scripts run only on the sensor. By default, they are installed in the directory */usr/local/SHADOW/sensor*.

4.2.1.1 sensor_init.sh

This script is placed into the */etc/rc.d/init.d* directory to start the sensor script at system boot up.

4.2.1.2 sensor_driver.pl

This script is the principal driver script for the sensor. It is executed hourly from the sensor *crontab* to cycle the *tcpdump* process in order to collect the packet data in hourly files. The script calls *stop_logger.pl* and *start_logger.pl* to stop the old collection process and start a new one.

4.2.1.3 stop_logger.pl

This script stops the *tcpdump* process started by a previous *start_logger.pl*. It is called by *sensor_driver.pl.*

4.2.1.4 start_logger.pl

This script starts a new *tcpdump* process. It is called by *sensor_driver.pl.*

4.2.1.5 std.ph

This script is a Perl header file that contains full path locations of commands used by the sensor scripts, full path locations of the saved *tcpdump* files, and other sensor configuration information. See section 5.1.8 for more information about configuring the sensor.

4.2.2 Analyzer

The following scripts run only on the analyzer. By default, the scripts are located in the */usr/local/SHADOW* directory.

4.2.2.1 fetchem.pl

This script is the principal worker of the SHADOW analysis system. Once per hour, it fetches a previous hour's data file from a sensor and copies it to the analyzer storage. Upon completing the file fetch, the script uncompresses the data file and feeds it to a *tcpdump* process for each filter in the analyzer's filter directory. The output from the *tcpdump* processes is sorted by IP address, condensed to reduce its bulk, and written in HTML to a web page. The script then runs *find_scan.pl* to look at the data file for evidence of scans. What evidence it finds is added to the end of the web page, and *fetchem.pl* completes.

4.2.2.2 cleanup.pl

This script is run each night to remove data files from a sensor's disks according to the specifications in the */etc/shadow.conf* configuration file. See section 5.2.7.1.

4.2.2.3 condense.pl

This script is run by *fetchem.pl* to take the output from multiple *tcpdump* output files to remove lines that are similar in content and replace them with a summary line listing the repeat count. This script cuts the sizes of the web pages dramatically.

4.2.2.4 obfuscate.pl

This script is called by the CGI script *compose_IR.cgi* to obfuscate the IP addresses of a local site to permit sharing of information with other intrusion detection or CERT organizations without releasing actual IP addresses to the public at large. In fact, the script is general purpose enough to obfuscate any text file that contains IP addresses, email addresses, or fully qualified domain names. It must be configured in the */etc/shadow.conf* configuration file. See section 5.2.7.1.

4.2.2.5 find_scan.pl

This script is called by *fetchem.pl* to look at an hourly file of *tcpdump* data and collect information about how many different "local" IP addresses and/or ports each external source address contacts. Although not a foolproof scan detector, it can point out potential scans that are not discernible from normal SHADOW web pages. The information is added to the end of each hour's web page.

4.2.2.6 comment_strip

This script is not called by any of the other scripts. It is provided to assist in the development of *tcpdump* filters. *Tcpdump* does not permit comments in its filter files. The filters used by SHADOW can consist of multiple files. This script allows an analyst to document his filters via comments and then strip off the comments before *fetchem.pl* accesses them from the SHADOW filter directory.

4.2.2.7 Statistics Scripts

New with version 1.8 of the SHADOW Package are scripts that are run to generate a daily page of IP statistics that are collected from the SHADOW data files. A "STATISTICS" button in the SHADOW tool window allows the SHADOW analyst to access previous days' statistics summaries. The scripts are located by default in the */usr/local/SHADOW/stats* directory.

[***Note:***] The statistics page is created only after all the twenty-four hourly files have been generated, i.e., once per day. So when selecting the particular statistics page to view, remember that the statistics for today have not yet been created. Yesterday's file is the most recent one available for viewing.

4.2.2.7.1 statistics.ph

This file is a Perl "header" file which contains a number of parameter and subroutine definitions, which are used to read the raw compressed *tcpdump* data files, accumulate IP statistics, store the statistical data in a file, and print the statistics.

4.2.2.7.2 do_daily_stats.pl

This script is responsible for generating a "*daily_stats.dat*" file in the directory in which the *tcpdump* data files are kept. Normally running late at night, it cycles through all 24 hourly files, and accumulates the statistics of interest in several Perl hash structures. The contents of those structures are saved in the "*daily_stats.dat"* file, created by the *Storable.pm* Perl module available from http://www.cpan.org/, for later access. The script also generates the "*stats.YYYYMMDD.html"* file that displays the collected statistics, and individual *html* files for each of the top local and top remote addresses in terms of traffic for that day.

4.2.2.7.3 print_stats.pl

This script is not run regularly by the SHADOW system. It is provided to read and print the information contained in a selected *Storable* file created by one of the other statistics scripts.

4.2.2.8 The CGI Scripts

The scripts in this section are CGI (Common Gateway Interface) scripts. They are run in response to user-initiated actions on the SHADOW web pages. See section 5.2.7.5 for information on configuring these scripts. Within the SHADOW distribution, the scripts are located by default in the */usr/local/SHADOW/httpd/cgi-bin* directory. Depending on how the Apache server is configured, they should also be located in the "ScriptAlias" directory as defined in the Apache configuration file. See section 5.2.6.1.

4.2.2.8.1 tools.cgi

When a web browser first accesses the SHADOW home page, another window pops up that is created and serviced by the *tools.cgi* script. This script presents buttons and links for activating other tools and displaying SHADOW pages by selecting sensor, date, and time.

4.2.2.8.2 search.cgi

Clicking the "SEARCH" button in the Tools window activates this script. An HTML form is displayed to permit the SHADOW analyst to search the *tcpdump* data files using date, time, host or network IP addresses, ports, and user-defined *tcpdump* filter expressions in arbitrary combinations. The script accepts the form input, searches the data, and presents the results on the web page. This allows analysts to "drill down" into events in order to display more details than would normally be displayed on the static web pages.

4.2.2.8.3 whois.cgi

Upon activation by selection of "WHOIS" button in the tools window, the *whois.cgi* script opens a window with a fill-in form to perform a "whois" inquiry to identify the individuals registered as responsible for administering a particular IP addresses or domain name.

4.2.2.8.4 lookup.cgi

Another button in the tools window activates the *lookup.cgi*. This script opens a window with a fill-in form to perform an "nslookup" inquiry that does a domain name to IP address translation.

4.2.2.8.5 compose_IR.cgi

This script opens a window with a fill-in form to allow the SHADOW analyst to create an Incident Report for reporting intrusion events to responsible computer incident handling teams and/or organizational management. Headings and labels for the form and reports are defined by parameters in */etc/shadow.conf*; see section 5.2.7.1. Execution of the scripts is initiated by selection of the "REPORT" button in the tools window.

4.2.2.8.6 kill_group.cgi

This script is not called directly. While executing a search for information in the *tcpdump* data via *search.cgi,* a click of the browser "stop" button stops the output from the script from reaching the browser window, but it does not stop the script executing on the web server. Since searching is such a CPU intensive process, several simultaneously running scripts can saturate a server. Consequently, a "kill" button was placed in the search window. Clicking on that button kills the search process and frees up the system resources.

4.2.2.8.7 nmap.cgi

The tools window has one more button, labeled "NMAP." Selection of this button calls the *nmap.cgi* script which presents the analyst with a fill-in form to perform a scan of a machine or group of machines to determine some of their operational status, including which TCP or UDP ports are open for connection. Based on knowledge of operating system responses to network stimuli, a reasonable guess as to the name and version of the operating system running on a target is made and reported back via the NMAP tool.

4.2.2.9 JavaScript

Many of the CGI scripts depend on small JavaScript functions to manipulate the browser windows in which the CGI scripts display their output. Within the SHADOW distribution, the JavaScript file is located by default in the */usr/local/SHADOW/httpd/js* directory. Depending on how the Apache server is configured, they should also be specified by an "Alias" entry in the Apache configuration file. See section 5.2.6.1.

4.2.2.9.1 openwin.js

This file contains six JavaScript functions used by the SHADOW CGI scripts. They are:

- OpenWindow(page, win_name, horiz, vert): which opens a window of size “horiz” X “vert”, with the window named “win_name”, accessing the URL specified by “page.”

- whoiswin(): which opens a window for the whois.cgi script.

- SearchWin(): which opens a window for the search.cgi script.

- ReportWin(): which opens a window for the compose_IR.cgi script.

- ToolWin(): which opens a window for the tools.cgi script.

- KillWin(): which opens a windo for the kill_group.cgi script.

5.0 Installation

Installation of SHADOW will be described in two sections, one for the sensor and one for the analyzer.

5.1 How to Build a SHADOW Sensor.

5.1.1 Sensor Hardware Requirements.

A SHADOW sensor does nothing but read IP packets from a network interface and write them to disk, so it requires only the hardware necessary to perform those tasks. It currently runs on systems with as little as 128MB of physical memory, although more memory always improves performance. A SCSI card and SCSI disks are highly recommended because they are faster than IDE drives. Experimentation will be required to determine the sizes and numbers of disks, based upon the volume of network traffic. The sensor usually holds the data files only for a short time, so huge disks are unnecessary unless the monitored network has very high traffic volume. To make the sensor externally invisible, two network cards are recommended; one with no IP address to collect the data, and the other which will be logically placed inside the site firewall for communication with the SHADOW analyzers. A single NIC can be used, but then the interface responsible for collecting network traffic is also responsible for transmitting the large data files, presenting possible conflicts and data loss. In addition, a single NIC will have to have an IP address to enable connection from the analyzers, thus making it visible and potentially more vulnerable from sources external to its network.

5.1.2 Install and Patch the Sensor Operating System.

This document is not a step-by-step guide to the installation of the Red Hat Linux operating system. The official Red Hat Installation guide, [1], suffices for that purpose. This document identifies points at which a divergence from a “normal” installation step, or a specific configuration choice should be made.

The first choice to be offered by the installation process is the “class” of installation. Red Hat offers five choices: workstation, server, laptop, custom, or upgrade. If an existing SHADOW sensor is being upgraded to a later version of the operating system, choose “upgrade.” Otherwise, choose “custom.” The sensor is a limited usage computer. It doesn't need much of the functionality that Red Hat includes for its workstation, server, or laptop classes. Using the custom class enables choosing the individual packages to be installed.

Shortly after beginning the install, the opportunity to partition the disks is presented. By default on Linux, `tcpdump` collects only 68 bytes of the IP packet. Still, a compressed file of one hour’s data can exceed 250 MB during the busiest times. For weekdays, the accumulated size of the data files can exceed 3 GBytes. That's a lot of data, so the partition on which the data will reside needs to be large enough to hold at least a day's worth. Disk and partition sizes should be adjusted as the amount of collected data changes.

During the network configuration portion of the installation process, the opportunity to configure the network interface cards (NICs) arrives. If the sensor has the two recommended NICs, only one of them should be configured with an IP address. One NIC of the sensor will be made “invisible;” it will have no

assigned address and will not be visible to other machines on the network segment to which it is connected.

The “Firewall Configuration” section of the installation process is intended to provide kernel level capability to filter, block, and/or log incoming IP packets. The same result is achieved by manually constructing an *iptables* script, as described in 5.1.10 and Appendix C:, so this installation step may be bypassed. Otherwise, choose the “High” security level. The sensors need to be protected as much as possible. However, for SHADOW, the analyzer must be able to connect to the sensor via SSH to fetch the data files and for periodic maintenance. Detailed configuration of SSH will be described in section 5.2.6.4 below.

Eventually, the installation process will reach the Package Group Selection section. A sensor needs no X Windows, Multimedia, Games, Sound Services, Development Tools, etc. If a lot of those tools are installed on the system, they will be available to potential intruders. Granted that the missing tools would only temporarily inconvenience a sophisticated intruder, that delay may provide enough extra time to detect and react to a system compromise. Some common sense can provide guidance about what packages to install. If the software is not directly related to the job of a SHADOW sensor, i.e. reading and storing network packet data, it's unnecessary. It can be installed later if a need arises. Since a custom kernel should be built for the sensor, (see section 5.1.3), the development tools can be left installed until after that custom kernel rebuild is done. A more secure alternative would be to build a kernel on the analyzer and transport it to the sensor. This process is highly recommended for installing new kernel releases on sensors.

After the distribution installation has completed and the system has re-booted, the web site http://updates.redhat.com/ will contain relevant updates in RPM format, (Red Hat Package Manager, the files in which Red Hat furnishes its software), for the installed version of the operating system. These are the “Patches” to the distribution that have been made since it was released. ***IT IS VERY IMPORTANT TO INSTALL THESE PATCHES AND TO PERIODICALLY MONITOR THE SITE FOR ADDITIONAL ONES!!!!*** Security risks are constantly changing; security preparedness is a never ending job. Someone out there is actively working on new exploits to attack unpatched systems. Once all the RPMs have been collected, the *rpm* command updates the system:

```
rpm -Fvh *.rpm
```

Updated RPMs for the kernel need not be fetched. The kernel will be built and customized to the sensor's hardware in step 5.1.3 below.

If two NIC cards were installed, the lines:

```
DEVICE=eth1
ONBOOT=no
```

should be the only contents of the */etc/sysconfig/network-scripts/ifcfg-eth1* file, where *eth1* is replaced by the name of the interface that will be invisible. This will insure that at each system boot, that interface will be brought up with no assigned address enabling *tcpdump* to capture the packets “invisibly.” This makes the sensor difficult to detect and attack. With no address, this interface cannot be the destination of any incoming packets.

5.1.3 Build a Custom Sensor Kernel.

As released, the Red Hat operating system includes a kernel that is configured to recognize and handle a large variety of hardware configurations. It is built with “kernel loadable modules”, (small pieces of code that the kernel can load and execute as needed), enabled. For example, if the machine has a 3Com Ethernet card, the boot process will detect it and load the appropriate 3Com driver module when it brings up networking. Any other Ethernet loadable modules will be ignored unless another brand of NIC is installed on the hardware. For maximum flexibility in the installation process, Red Hat provides a large number of kernel loadable modules for many of the most likely hardware configurations. This greatly simplifies initial OS installation. As the boot process proceeds, hardware components are recognized, and the kernel loads appropriate drivers for those components. In effect, the kernel dynamically reconfigures itself to match the hardware it finds.

Of course, there are downsides to this flexibility. The released kernel is configured to use only those machine language instructions common to all x86 Intel processors, i.e. the 80386. It is not optimized to avail itself of extended instructions implemented only on later processors in the Pentium series. Many modules are included for hardware configurations that will never exist on one particular system. Kernel loadable modules also present security risks. It has been documented that the creation of a “Trojan” kernel module is possible. So, for example, a module could be created to give special privileges to a particular process if the kernel loads the module into memory. If this module is given the name of a hardware driver, the kernel can be signaled to load it. When that happens, the creator of the module has a process running with kernel privileges.

To avoid those problems and to optimize the performance of the Linux kernel for specific hardware, a custom kernel should be built. The latest version of the Linux kernel can be downloaded from http://www.kernel.org to insure that it has the latest fixes and capabilities. Here are the steps to build a custom kernel:

```
1. Download the latest kernel, e.g. linux-{VERSION}.tar.gz.
2. cd /usr/src; tar xvfz /some/path/linux-{VERSION}.tar.gz
3. cd /usr/src/linux-{VERSION}
4. make menuconfig(or xconfig if running X Windows)
```

At this point, a menu appears that allows for the customization of possible hardware and software configurations. A 'N' answer should be given for all drivers for hardware not on the system. A 'N' answer should be given for all software unnecessary for the sensor, e.g. development tools, sound support, X-windows, documentation, etc. A 'N' answer should be given to “Enable loadable module support.” When the configuration process is complete, the “Save and Exit” button will cause the creation of a file, “*.config*” containing all the configuration options selected during the process. Appendix A: contains a copy of a sensor *.config* file that defines the parameters used in a custom kernel build. Once the configuration file matches the system hardware and desired software, the new kernel is built and installed by:

```
1. cd /usr/src/linux-{VERSION}
2. make dep
3. make bzImage
4. cp System.map /boot/System.map-{VERSION}
5. cp arch/i386/boot/bzImage /boot/vmlinuz-{VERSION}
6. Edit the file /etc/lilo.conf or /etc/boot/grub/grub.conf to include the new kernel just built.
7. If using LILO as the boot loader, run /sbin/lilo to re-install the boot record with the new kernel
information.
```

```
8. Re-boot the system to see if the new kernel will run.

Where {VERSION} is the version of the kernel being built, e.g. 2.4.19.
```

If the build process fails, the kernel may additional reconfiguration. Tips for kernel building may be found at: http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html.

5.1.4 Create the SHADOW User Account on the Sensor.

On the SHADOW analyzer, a periodic process runs to fetch data files from the sensor. In compliance with the rule of "least privilege", it is unnecessary and risky for that analyzer process to have root level privileges on the sensor. So, an unprivileged account should be created on the sensor, to correspond to one created on the analyzer, (see 5.2.4), as owner of the transfer process. The name "shadow" is an example user created for this purpose. To create the "shadow" user. As root:

```
useradd -c "Lamont Cranston, the SHADOW" -u 666 -d /home/SHADOW shadow
```

By default, Red Hat Linux automatically creates a group with the same name as the user. **Do not give "shadow" a password, so no one can log in directly at the console as "shadow."** Later, in section 5.2.6.4 "shadow" will be given some *OpenSSH* keys to use for communication with the sensors. At that point, the *OpenSSH* public keys will be copied to the sensor, to permit the analyzer "shadow" user to activate processes on the sensor as user "shadow".

5.1.5 Unpack the SHADOW-1.8 Software.

The SHADOW software is released in a compressed tarball: SHADOW-1.8.tar.gz. Choose a location for unpacking the software. "*/usr/local/SHADOW*" might be a good place:

```
1. mkdir -p /usr/local/SHADOW
2. cd /usr/local/SHADOW
3. tar xvfz /tmp/SHADOW-1.8.tar.gz
```

5.1.6 Install *Tcpdump* on the Sensor.

With release 6.2 and later of Red Hat Linux, Red Hat released a modified version of the *tcpdump* program from Lawrence Berkeley Labs. They applied a number of patches that extended its capabilities, but also changed its output format. The SHADOW scripts parse the output from *tcpdump*, so they will fail with these Red Hat versions of *tcpdump*. A working version of *tcpdump*, (Version 3.7.2), is furnished in the SHADOW package. It is a copy of the latest version from http://www.tcpdump.org.

Install the *tcpdump* included in the SHADOW package with the following command:

```
cd /usr/local/SHADOW/accessories/rpms
rpm -Uvh tcpdump-3.7.2-wdr3.i386.rpm
```

The source tarball and source RPM files are also furnished, so *tcpdump* can be rebuilt from source. Also included in the source distribution is an RPM for the corresponding version of *libpcap*, the packet capture library on which *tcpdump* is based. It is not necessary for SHADOW, but is included as a surprise bonus for those who may want to build their own packet capture software.

5.1.7 Install OpenSSH on the Sensor.

The analyzer fetches files from the sensor. The mechanism chosen to implement that communication path is *OpenSSH*, the secure shell. *OpenSSH* provides a method for communication between analyzer and sensor that enhances authentication and is resistant to eavesdropping. Red Hat Linux 8.0 furnishes version 3.4p1 *of OpenSSH* in its distribution in five RPMs:

- *openssh*
- *openssh-server*
- *openssh-clients*
- *openssh-askpass*
- *openssh-askpass-gnome*

Included in the accessories sub-directory of the SHADOW package is *OpenSSH* version 3.6.1p1 in a single RPM. Install it by:

```
cd /usr/local/SHADOW/accessories/rpms
rpm -Uvh openssh-3.6.1p1-wdr1.i386.rpm
```

The tarball and SRPM for OpenSSH 3.6.1p1 are included as well.

Previous versions of SHADOW recommended that an unprivileged user on the analyzer be permitted to log in as root on the sensor to fetch the files. For moving files, it is unnecessary and risky to allow root access. Consequently, an ordinary user account, e.g. “shadow”, should be created on the sensor to be the owner of the process that transports the *tcpdump* data files. See section 5.1.4 for more details.

For OpenSSH to work correctly, the sensor file */home/shadow/.ssh/authorized_keys* must contain copies of the SSH public keys of the *shadow* user that were generated on the analyzer. Detailed instructions about installing SSH between the analyzer and the sensors are provided in the section 5.2.6.4. The book cited in Ref. [2] is highly recommended; it's recognized as the best available reference for understanding SSH's workings and how to configure it. Configuring *OpenSSH* can be very tricky until the necessary experience is acquired.

5.1.8 Configure SHADOW Sensor Software.

After the software package is unpacked, it needs to be configured. Three files require customization: *sensor_init.sh, std.ph, and std.filter*. Here are the requirements:

1. For sensor_init.sh:
   (a) Set the variable SENSOR_PATH to point to where the SHADOW package is installed, e.g. /usr/local/SHADOW/sensor.
   (b) Set the variable, SENSOR_PARAMETER to the name of the Perl header file which contains the configuration parameters, e.g. std (leave off the .ph extension).

2. For std.ph (or the equivalent for a specific installation):
   (a) Set the variable $LOGPROG to point to the location of the *tcpdump* binary.
   (b) Set the variable $PROGPAR to indicate the interface which will be used to capture passing IP packets. It will be set to “-i eth1” if eth1 is the “invisible” interface.

(c) Set the variable $LOGFILE to point to the directory into which the *tcpdump* data files will be written. These files are potentially very large, so make sure there is plenty of room. "/LOG" is the default.
(d) Set the variable, $FETCHEM_USER to the name of the user account created to be the owner of the data files, e.g. "shadow".

3. For *std.filter*, define a *tcpdump* filter to insure that all the traffic needed for viewing and searching is captured from the invisible interface. The s*td.filter* file furnished in the SHADOW-1.8 package contains the single line: "ip." This filter says capture all IP packets. This filter may be modified to eliminate some traffic unwanted in the data files. Make this a simple filter so that the *tcpdump* program does not have to work too hard deciding which packets to save. More sophisticated filters are defined for the analyzer machine to highlight "interesting" traffic. See section 5.2.7.3. A large part of SHADOW's value comes from the fact that the raw data files contain the IP packet headers for all the traffic that passed by the sensor. This permits later searches for traffic patterns unanticipated at the time of sensor installation. For example, if the sensor filter excludes SMTP traffic, then future analysis of mail traffic is impossible because that data was not stored in the raw files.

5.1.9 Configure Crontab and System Start-up Files for the Sensor.

The script *sensor_driver.pl* is the main controlling process of the sensor. It runs once per hour to end the previous hour's *tcpdump* process and start the next one. The process is started by the system *crontab*, a mechanism under Unix/Linux for starting processes at specific times. The SHADOW release package contains a file */usr/local/SHADOW-1.8/sensor/sensor_crontab* which contains:

```
 1  #
 2  # SHADOW Release 1.8
 3  # Last Modified 3 Jul 2001
 4  #
 5  # Written by Bill Ralph <RalphWD@nswc.navy.mil>
 6  #
 7  # Crontab for a tcpdump Sensor.
 8  #
 9  # Insure system clock is consistent.
10  # If this is not on the internet, need a standard time source.
11  #
12  17 23 * * * /usr/bin/ntpdate time-server.righteousdudes.com
13  #
14  # The next command insures that the hardware clock of a PC is reset
15  # to the time that the previous command set the Operating System
16  # time to. Only meaningful for Linux or Intel x86 systems perhaps.
17  #
18  18 23 * * * /sbin/hwclock -systohc
19  #
20  # The startup of the SHADOW sensor driver, once per hour.
21  #
22  0 * * * * /usr/local/SHADOW/sensor/sensor_driver.pl std > /dev/null 2>&1
```

Line 12 uses a Network Time Protocol (NTP) process (*ntpdate),* at 11:17 PM each evening to fetch the current time from a reliable local or stratum-2 timeserver and set the system clock to that time. See http://www.ntp.org/ for information on NTP and the available publicly accessible timeservers. The sensor and analyzer should have reasonably close ideas of what time it is. In fact, it's useful to have the site firewall synchronized to the same time. Line 18 resets the sensor's hardware clock to the value of the

system clock at 11:18 PM each evening. Line 22 starts the *sensor_driver.pl* script at the zero-th minute past each hour.

After site customization, the *sensor_init.sh* script in the distribution should be moved. The variable SHADOW_PATH in that file needs to be set to the location into which the SHADOW package was loaded. Then, copy that file into the */etc/rc.d/init.d* subdirectory with:

```
1.cd /usr/local/SHADOW/sensor
2.cp sensor_init.sh /etc/rc.d/init.d/sensor
3.chkconfig --add sensor
```

This will insure that the SHADOW *tcpdump* process will restart after each system re-boot.

5.1.10 Protect the Sensor.

Strip the system of all network services except SSH. On Red Hat Linux 8.0, this involves removing all services from */etc/xinetd.conf* and either removing all the files in the */etc/xinetd.d* directory, or adding the line "`disable = yes`" to each. The command "`chkconfig --list`" will list all daemons that are started at bootup as well as all services that are started by `xinetd`. The *tcp_wrappers* system furnished with Red Hat Linux is an additional security mechanism. *Tcp_wrappers* allows the definition of access control lists for TCP/IP services or ports that are open on a system. The only TCP/IP service that sensor needs is the SSH daemon to which the analyzer will connect, so the */etc/hosts.allow* file should contain something like the following, where `172.16.91.98` is replaced with the IP address of the sensor, (the sensor allows all TCP/IP services to connect from itself), and 172.16.222. is replaced by the network address of the subnet on which the SHADOW analyzers are located:

```
#
# hosts.allow        This file describes the names of the hosts which are
#                    allowed to use the local INET services, as decided
#                    by the '/usr/sbin/tcpd' server.
#
ALL:            172.16.91.98,127.0.0.1          : severity auth.info:        allow
sshd:           172.16.222.                      : severity auth.info:       allow
sshdfwd-X11:    172.16.222.                      : severity auth.info:       allow
ALL:            ALL                              : severity auth.info:       deny
```

See the man pages for *hosts.allow* and *hosts_options.*

Like any networked computer, the sensor may be vulnerable to compromise. It should not be easy to compromise, and even if it is compromised, the machine certainly reveal secrets about the rest of network. In the SHADOW release package, and included as Appendix C: is an example *iptables* file which may be used to provide additional protection for SHADOW systems. The first part should be customized with the name of the active Ethernet interface, and the IP addresses of the analyzer(s) that will connect. The script allows IP communications on the SSH port between the analyzer(s) and the sensor, and only those other services that may be necessary, such as NTP. It blocks and logs all other attempts to connect to the sensor. It also adds a bit of insurance to the invisibility of the passive Ethernet interfaces, in that it prohibits ANY packets from being sent out that interface.

5.1.11 Situate the Sensor.

At this point, the sensor is configured until the analyzer is ready to connect. The sensor should be situated at a position in the network at which IP traffic is to be captured. The invisible interface should be connected to a hub or the span port of a switch outside the firewall to see who is knocking at the door, or directly inside to see who got through the door. It's sensible to station two sensors, one outside and one inside to measure the effectiveness of the firewall.

> [***Note:***] If the sensor is connected to a regular switch port, no traffic except broadcast traffic on the network segment on which the invisible interface is placed will be seen. Consult with local networking expertise for help in setting up a “span”port to allow promiscuous collection of IP packets.

The sensor is ready; the next step is to build the analyzer.

5.2 How to Build a SHADOW Analyzer.

5.2.1 Analyzer Hardware Requirements.

Here's a review of the SHADOW system architecture. The sensor sits near the interface between a LAN and the Internet. It captures network packets to disk. The sensor has been hardened so that it is hard to compromise. However, if it is compromised, little information about the internal network and few tools are left on the sensor to assist the successful intruders. The sensor is untrusted, so a compromise won't provide a trusted path to the internal network. The sensor is configured to accept only SSH connections and only from the analyzer. The analyzer is the workhorse of the SHADOW system, scripts running on the analyzer initiate connections to the sensor, retrieve data files from the sensor, and clean up the sensor's disk space. The analyzer filters the data files to produce web pages populated with “events of interest,” serves those web pages via Apache, offers analysis tools to retrieve more information about the events, and provides an interface for building incident reports. The faster and more powerful the analyzer is, the better.

> [***Note:***]The question is frequently asked about building a single computer to act as both sensor and analyzer. It's possible, but strongly discouraged. There are at least two problematical areas of concern, security and performance. From a security aspect, an intrusion detection system is supposed to enhance, not reduce the security of a network. The analyzer has a lot of installed software beyond what a sensor needs, and it may have trust relationships with other machines inside the firewall. With separate sensors and analyzers, a compromised sensor adds very little security risk. With a combined sensor/analyzer, security risks are greater. From the performance aspect, a machine which does the searching and serving tasks of an analyzer while simultaneously capturing every packet header, may falter under the load.

The CPU should be the fastest possible. Machines with ratings of 350-1600 MHz have been used with great success. Of course, faster machines perform better, particularly if the data files are huge. The more memory the analyzer has, the better. Fast ample disk storage is required to hold the data files. If keeping the data available for a time is important, it would be worthwhile to construct a RAID system to hold it. The data access should be as fast as possible; *tcpdump* data searches are extremely I/O intensive.

The analyzer system may be a full-featured workstation, with sophisticated video, sound, and all the bells and whistles, if it is to be used as an analysis workstation. It may be configured as a web server only, so, for example, a high performance video card in the analyzer would be unnecessary. It is also possible to split the analyzer work over multiple machines, e.g. one driving a RAID, one running the web server, one fetching the data, and one doing the searches. This is all vague; the specifications for the analyzer depend on the amount of data collected, how long it is to be held on disk, how many people need access to the SHADOW web pages, and how many data searches are conducted. The cardinal rule for the analyzer hardware is: the faster the better!

5.2.2 Install and Patch the Analyzer Operating System.

Be sure to read section 5.1.2. The “custom” configuration option during the Red Hat installation process is the best choice for the SHADOW analyzer just as it was for the sensor. It gives better control over which packages are installed by default.

During the partition dialog, keep the following in mind. The analyzer fetches data files from the sensors, uses the data to generate web pages, and saves the data files for later use in searches. That means that the analyzer must hold the data files from all the sensors for as long as it is desired to search them. Keep in mind the points made in section 5.2.1, the analyzer requires fast and commodious disk resources. How the disk is partitioned is a matter of personal preference, given the analyzer requirements, the amount of data to be stored, and the numbers and sizes of the disks.

When the Red Hat install procedure asks about selecting packages to install, make sure to include *apache* and *iptables*. Be sure to verify that *Perl* and the development tools are also included. Packages containing risky stuff that shouldn't be used on any system for security reasons can be excluded: *rsh-server*, *rusers-server*, *wu-ftpd*, *tftp-server*, *telnet-server*, etc.

After the operating system and basic utilities are installed, go to Red Hat Updates http://updates.redhat.rpm/ and download the updated RPMs (Red Hat Package Manager, the files in which Red Hat furnishes its software), for the installed version of the OS. These are the patches, including security fixes, to the system and utilities that have been made since the distribution was released. ***IT IS VERY IMPORTANT TO INSTALL THESE FIXES AND TO FREQUENTLY MONITOR THE SITE FOR ADDITIONAL ONES!!!!*** Security risks are constantly changing; security preparedness is a never-ending job. Someone out there is actively working on new exploits to attack unpatched systems. Once all the RPMs have been collected, the *rpm* command updates the system:

```
rpm -Fvh *.rpm
```

Updated RPMs for the kernel need not be fetched. The kernel will be built and customized in step 5.2.3 below.

5.2.3 Build a Custom Analyzer Kernel.

In step 5.1.3 above, a custom kernel was built for the sensor. The objective was to configure it such that only those kernel components that are absolutely necessary on the sensor are included. The operating system on the sensor is not expected to do much; keep *tcpdump* running to collect and save the data files. The analyzer is a different story. It fetches the data and runs scripts to produce and serve web pages. It also provides tools to allow searches of the data, searches of Internet databases, generation of incident reports, and maintenance operations on the sensors. Given the analyzer’s tasks, and a sensible security posture, it makes sense to optimize the kernel for the installed hardware and software.

> [***Note:***] This section is nearly a duplicate of 5.1.3. The process is the same; some of the selections will be different.

The latest version of the Linux kernel can be downloaded from Kernel.org http://www.kernel.org. Here are the steps to build a custom kernel:

```
1. Download the latest kernel, e.g. linux-{VERSION}.tar.gz.
2. cd /usr/src; tar xvfz /some/path/linux-{VERSION}.tar.gz
3. cd /usr/src/linux-{VERSION}
4. make menuconfig (or make xconfig if using X Windows)
```

At this point, a menu will offer all the possible hardware and software configurations that can be included in the kernel. Answer 'N' to drivers for hardware not on the system. Answer 'N' to all software unnecessary for a SHADOW analyzer, e.g. amateur radio, telephony, ISDN, etc. When the configuration process is complete, click on “Exit” (Menuconfig) or “Save and Exit” (Xconfig), which creates a file “*.config”* containing all the configuration options selected during the process. See Appendix B: for a copy of a *.config* file that specifies the kernel configuration for a typical analyzer. Once the configuration file matches the hardware and desired software, build the new kernel and put it into place by:

```
1.  cd /usr/src/linux-{VERSION}
2.  make dep
3.  make bzImage
4.  make modules
5.  make modules_install
6.  cp System.map /boot/System.map-{VERSION}
7.  cp arch/i386/boot/bzImage /boot/vmlinuz-{VERSION}
8.  If using LILO, edit the file /etc/lilo.conf to include the new kernel just built. If using grub, edit
/boot/grub/grub.conf.
9.  If using LILO, run /sbin/lilo to re-install the boot loader with the new kernel information.
10.  Re-boot the system to see if the new kernel will run.
11.  Where {VERSION} is the version of the kernel being built, e.g. 2.4.19.
```

Be sure to study the document on how to build a Linux kernel, which can be found at: Kernel-HOWTO http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html. It may take a few iterations to configure the kernel so that it compiles, boots, and includes all the desired pieces.

5.2.4 Create a SHADOW User Account on the Analyzer.

On the analyzer, the Apache web server should not run as a privileged user, namely “root”. A user account should be created as the owner of the SHADOW data files and the web tree containing the HTML files generated by the SHADOW scripts. A separate user account should be created to be the owner of the Apache *httpd* processes. More details are spelled out in the section 5.2.6.1. For the owner of the *httpd* processes, the default Apache installation creates the user “apache”. For the owner of the data and HTML files, the name “shadow” seems to fit, corresponding to the one created on the sensor in step 5.1.4 above. By separating the owner of the data files from the owner of the processes, a user who subverts an *httpd* process cannot freely alter the files, since he doesn't own them. To create the “shadow” user, as root:

```
useradd -c "Lamont Cranston, the SHADOW" -u 666 -m shadow
```

By default, Red Hat Linux automatically creates a group with the same name as the user. Do not give “shadow” a password. No one should be able to log in directly to the analyzer as “shadow.” In section 5.2.6.4 “shadow” will be given *OpenSSH* keys to use for communication with the sensors.

5.2.5 Unpack the SHADOW-1.8 Software

The SHADOW software comes in a compressed tarball: SHADOW-1.8.tar.gz. Choose a location for unpacking the software. “*/usr/local/*SHADOW” might be a good place:

```
1. mkdir -p /usr/local/SHADOW
2. cd /usr/local/SHADOW
3. tar xvfz /wherever/the/tarball/was/SHADOW-1.8.tar.gz
```

5.2.6 Install the Accessories.

Inside the SHADOW-1.8 distribution directory is a subdirectory "accessories" which contains pre-packaged accessories on which SHADOW depends. They are collected in subdirectories *RPMS, tarballs,* and *SRPMS*. The accessories included in this version of SHADOW are *tcpdump* Version 3.7.2 and *OpenSSH* Version 3.6.1p1. In Red Hat Linux Version 8.0, the version of *tcpdump* included was 3.6.3, and the version of *OpenSSH* was 3.4p1. The version of OpenSSH included in the accessories subdirectory is for convenience; the one supplied with the OS distribution will work. However, be aware that RedHat maintains its own *tcpdump* program, different than the one at *http://www.tcpdump.org.* The included *tcpdump* program is more compatible with the "standard" as released by Lawrence Berkeley Labs and maintained by *tcpdump.org*. The following subsections provide details about accessory installation.

5.2.6.1 Install and Configure Apache.

Apache is included in the Red Hat distribution. It should be installed with the rest of the system if "web server" was selected during the installation process. The *mod_ssl* and *mod_perl* modules may be installed as well to assist Apache. *Mod_ssl* adds Secure Sockets Layer processing, and *mod_perl* adds Perl processing to the Apache daemon itself, thus speeding up the execution of CGI scripts written in Perl.

Apache needs some SHADOW specific configuration. In the */usr/local/SHADOW/httpd/apache-conf* directory are files *httpd-1.3.XX.conf*, and *httpd-2.0.40.con.* These files are Apache configurations for the 1.3 and 2.0 versions of the server software. An additional file, *httpd-2.0.40.basic,* contains just the specific lines that pertain to the SHADOW specific configuration. That file contains (without the numbers):

```
1       Timeout 36000
2       KeepAlive On
3       MaxKeepAliveRequests 100
4       KeepAliveTimeout 15
5       Listen 80
6       ExtendedStatus Off
7       User apache
8       Group apache
9       ServerAdmin admin@righteousdudes.com
10      ServerName shadow-analyzer.righteousdudes.com:80
11      UseCanonicalName On
12      DocumentRoot "/home/shadow/html"

13      <Directory "/home/shadow/html">
14      Options Indexes Includes FollowSymLinks
15      AllowOverride All
16      AuthType Basic
17      AuthName "Authorized SHADOW Users"
18      AuthUserFile /usr/local/SHADOW/Authorization/SHADOW.pwd
19      Satisfy any
20      require valid-user
21      Order deny,allow
22      Deny from all
23      Allow from 172.21.122
24      Allow from localhost
25      Allow from 127.0.0.1
```

```
26      </Directory>

27      HostnameLookups Off
28      ServerSignature On
29      Alias /icons/ "/home/shadow/html/icons/"
30      Alias /images/ "/home/shadow/html/images/"
31      Alias /js/ "/home/shadow/httpd/js/"

32      <Directory "/home/shadow/html/icons">
33      Options Indexes MultiViews
34      AllowOverride None
35      Order deny,allow
36      Deny from all
37      Allow from 172.21
38      Allow from .righteousdudes.com
39      Allow from localhost
40      Allow from 127.0.0.1
41      </Directory>

42      <Directory "/home/shadow/html/images">
43      Options Indexes MultiViews
44      AllowOverride None
45      Order deny,allow
46      Deny from all
47      Allow from 172.21
48      Allow from .righteousdudes.com
49      Allow from localhost
50      Allow from 127.0.0.1
51      </Directory>

52      <Directory "/home/shadow/httpd/js">
53      AllowOverride All
54      Options +FollowSymLinks
55      Order deny,allow
56      Deny from all
57      Allow from 172.21
58      Allow from .righteousdudes.com
59      Allow from localhost
60      Allow from 127.0.0.1
61      </Directory>

62      ScriptAlias /cgi-bin/ "/home/shadow/httpd/cgi-bin/"
63      ScriptAlias /privileged/ "/home/shadow/httpd/cgi-bin/privileged/"

64      <Directory /home/shadow/httpd/cgi-bin>
65      AllowOverride All
66      Options ExecCGI FollowSymLinks
67      Order deny,allow
68      Deny from all
69      Allow from 172.21
70      Allow from .righteousdudes.com
71      Allow from localhost
72      Allow from 127.0.0.1
73      </Directory>

74      <Directory "/home/shadow/httpd/cgi-bin/privileged">
75      AllowOverride All
76      Options ExecCGI FollowSymLinks
77      AuthType Basic
78      AuthName "Privileged SHADOW Users"
79      AuthUserFile /home/shadow/httpd/cgi-bin/privileged/priv.pwd
80      Satisfy any
81      require valid-user
82      order deny,allow
83      deny from all
84      allow from 172.21.122
85      </Directory>
```

Line 1 sets the client timeout to 10 hours, in contrast to the default 360 seconds. When SHADOW performs a search, the process of unzipping the data files and feeding them to *tcpdump* can take longer than 6 minutes. Lines 2-11 set some global parameters, including the definition of the user account,

('apache'), under which the httpd processes run. Line 12 and lines 13-26 define the location of the root documents of the HTML tree that Apache is serving, and specify the access control defining who can access those documents. In this case, all users from 172.21.122 are allowed access, as well as anyone else who has been granted a password stored in the file */usr/local/SHADOW/Authorization/SHADOW.pwd.* This policy insures that an organization's network traffic data is not available to everyone who can direct his browser to the SHADOW analyzer. Reference [10] provides more details about establishing a "Basic Authorization" policy for Apache.

Lines 29-61 define three aliases: one for the directory in which SHADOW icons are located, one for the SHADOW specific images directory, and one for the location of JavaScript functions that SHADOW uses. As well as defining the absolute locations of those directories, this section of configuration defines the optional features of each directory and the access control rules for those allowed to access files from each. In the cited example, access is granted to everyone on the local network, 172.21 or *righteousdudes.com*, and the localhost. Lines 62-85 accomplish similar rules for the script aliases: */cgi-bin* and */privileged* where the executable CGI scripts are to be located. The */privileged* alias was created to require additional authorization for the *compose_IR.cgi* and *nmap.cgi* scripts. Not everyone who can view an organization's SHADOW data should be able to submit an incident report or perform an *NMAP* scan of a machine. The directives in this section provide a mechanism for specifying who can perform those functions.

See Ref. [10] and http://www.apache.org/ for more details on configuring the Apache server and the meanings of the directives listed above.

5.2.6.2 Install the SHADOW Home Web Page.

The *httpd.conf* file installed in section 5.2.6.1, by default, expects the SHADOW home page to be located in */home/shadow/html*. Create the home page by:

```
1. mkdir -p /home/shadow/html/tcpdump_results
2. cd /usr/local/SHADOW/httpd/home
3. cp * /home/shadow/html
4. cp .htaccess /home/shadow/html
5. chown -R shadow:shadow /home/shadow
```

5.2.6.3 Install *Tcpdump* on the Analyzer.

See the section 5.1.6 for information about the need to install a different version of *tcpdump* than the one furnished by Red Hat.

Install *tcpdump* with the following command:

```
cd /usr/local/SHADOW/accessories/rpms
rpm -Uvh tcpdump-3.7.2-wdr3.i386.rpm
```

The SRPM and tarball files are also furnished, so the binaries can be built from source.

5.2.6.4 Install and Configure OpenSSH.

SHADOW requires a communications path between the analyzer and the sensor, to fetch the data files, and perform periodic maintenance. Because the sensor may be located in a DMZ, outside the shield of the site firewall, that communication path may require some protection. Otherwise, anyone able to plant a sniffer in the DMZ would be able to pick up passwords for accounts on the sensor. And because the data files moved by the analyzer are snapshots of the site network traffic, it would be risky to transport that data in the clear. SSH was the mechanism chosen to solve to build a protected communication path. The SSH protocol encrypts the traffic between communicating machines using sophisticated battle-proven Open Source encryption algorithms. It also adds an additional layer of authentication between client and server to protect against spoofing. *OpenSSH* is the chosen SSH implementation for SHADOW. More information and source can be found at: http://www.openssh.org. Included in the SHADOW package is *OpenSSH* version 3.6.1p1 in a single rpm. Install it by:

```
cd /usr/local/SHADOW/accessories/rpms
rpm -Uvh openssh-3.6.1p1-wdr1.i386.rpm
```

If it balks, uninstall the version of OpenSSH included with Red Hat Linux:

```
rpm -e openssh openssh-server openssh-clients openssh-askpass
rpm -e openssh-askpass-gnome
```

As part of the installation, *OpenSSH* generates three host keys in its configuration directory */etc/ssh*: *ssh_host_key*, *ssh_host_rsa_key*, and *ssh_host_dsa_key*. These are the host's private keys; the public key counterparts are *ssh_host_key.pub*, *ssh_host_rsa_key.pub*, and *ssh_host_dsa_key.pub*, respectively. The *ssh_host_key* pair is for SSH protocol version 1 compatibility, and is largely deprecated now because of identified and publicized SSH1 security problems. The key pairs *ssh_host_rsa_key* and *ssh_host_dsa_key* are for SSH protocol version 2, which has been found to be more robust than version 1. The *ssh_host_rsa_key* is generated using the RSA algorithm and *ssh_host_dsa_key* was generated using the DSA algorithm. See [2]. The public halves of the host keys on the sensors need to be copied to the */etc/ssh/ssh_known_hosts* file on the analyzer as follows: (There are other mechanisms for getting the sensors' public host keys to the analyzer, this is one example.)

1. On the sensor, insert a floppy disk into the floppy drive.
2. As root, type “mount -t vfat /dev/fd0 /mnt/floppy”
3. cd /etc/ssh
4. cat ssh_host_key.pub ssh_host_dsa_key.pub ssh_host_rsa_key.pub > /mnt/floppy/ssh_known_hosts
5. umount /mnt/floppy

On the analyzer:

1. Insert the floppy.
2. As root, mount -t vfat /dev/fd0 /mnt/floppy
3. cd /etc/ssh
4. cp /mnt/floppy/ssh_known_hosts .
5. Umount /mnt/floppy

[**Note:**] Here's a quick description of why this is needed. As part of the SSH protocol, an incoming connection request is received by the sensor's *sshd* daemon which then transmits a copy of the sensor's public key to the originator of the connection request, the analyzer. The analyzer compares the public key he receives from the sensor to a stored value kept in its /etc/ssh/ssh_known_hosts file. If they match, the analyzer is sure he is connected to the

correct machine. Otherwise, someone could be attempting to impersonate the sensor or to intercept communications between the sensor and analyzer. This mechanism, called “public key authentication,” decreases the likelihood of that happening.

In section 5.2.4 above, a user account named “shadow” was created to be the owner of the processes and files within the SHADOW and Apache systems. The “shadow” account now needs to generate a private/public key pair so that it can connect from the analyzer to the sensors to fetch the data files. Generation of shadow's keys is done by:

```
1. Become root on the analyzer then “su - shadow” to become the “shadow” user.
2. mkdir .ssh
3. chmod 700 .ssh
4. cd .ssh
5. /usr/bin/ssh-keygen -b 1024 -t dsa -f .id_dsa.
When prompted for a passphrase, enter return and repeat.
6. /usr/bin/ssh-keygen -b 1024 -t rsa2 -f id_rsa2.
When prompted for a passphrase, enter return and repeat.
7. cat id_rsa1.pub cat id_dsa.pub id_rsa2.pub > authorized_keys
8. Insert a DOS formatted floppy.
9. mount -t vfat/dev/fd0 /mnt/floppy
10. cat id_rsa1.pub id_dsa.pub id_rsa2.pub > /mnt/floppy/authorized_keys
11. umount /mnt/floppy
```

Steps 5 and 6 above generate SSH keys for user shadow. Step 5 generates an SSH protocol 2 password using the DSA algorithm; and step 6 generates an SSH protocol 2 password using the RSA algorithm. Note that there is no passphrase on the shadow's private keys. As will be shown in section 5.2.6.5, the SHADOW scripts will be run as user shadow from an automated *crontab* script each hour. If the shadow private SSH key has a passphrase defined, then the automated script will halt to await entry of that passphrase to complete each connection from the analyzer to the sensor. By defining an empty passphrase, that situation is avoided, at a small additional risk. Creating the user shadow without the ability for anyone to log in directly as shadow, lessens that risk somewhat. To move the newly generated keys to the sensor:

```
1. Insert the floppy.
2. mount -t vfat /dev/fd0 /mnt/floppy
3. cd /home/shadow/.ssh
4. cp /mnt/floppy/authorized_keys .
5. Umount /mnt/floppy
```

SSH is particularly picky about permissions on its files and directories. Make sure that the *.ssh* directories, and all files contained in the directories, */home/shadow/.ssh* are owned by shadow and have permissions 700 on the directories and 600 on the files, except for the *“.pub”* files. In step 5.2.6.5 below, the SSH connection between the analyzer and the sensors will be tested to insure that the OpenSSH software is configured and operating correctly.

[***Note:***]Here's a high level explanation of the use of “shadow's” SSH keys. On the sensor, the file */home/shadow/.ssh/ssh_authorized_keys* contains copies of the public keys of all users allowed to connect as shadow to the sensor. Remember that private/public key encryption depends on the fact that a message encrypted with one half of a key pair can *ONLY* be decrypted by the other half of that pair. So if the sensor can decrypt a message using shadow's public key in the *ssh_authorized_keys* file, it could only have been encrypted by the corresponding private key, the one belonging to user shadow on the analyzer.

Once again, further clarification of the SSH process can be obtained in the book cited in Ref. [2]

5.2.6.5 Test the Analyzer to Sensor Connection.

Let's review. At this stage in the installation process, a two-computer SHADOW system is under construction, consisting of a sensor and an analyzer. The shadow account on the analyzer will be set up to automatically run an hourly script, (see section 5.2.9), to connect to the sensor and fetch the previous hour's compressed *tcpdump* data file, then run some filters on the data and generate a web page of the "interesting" events. Obviously, the crucial aspects of this relationship are the ability of the analyzer to connect to the sensor, fetch a file, and run some commands on the sensor. OpenSSH components have been installed on both machines. A "shadow" user account has been created on the analyzer. Some passphrase-free SSH keys for that account have been generated, with the public keys copied to the sensor. Now before the SHADOW system can become functional, the connection from the shadow account on the analyzer to the sensor must be tested and verified operational.

1. Become *root* on the analyzer; then type: "`su - shadow`" to become shadow.

2. Type: "`ssh -v sensor01.righteousdudes.com`", where sensor1.righteousdudes.com is the name of the sensor to which to connect. The -v option will produce a lot of debugging information, verbosely documenting each step through the SSH protocol. Where it finds problems, it will print messages that should help in solving them. If at the end of the dialog, the sensor's shell prompt is displayed, the ability of the analyzer to connect to the sensor has been verified. If not, proceed with the next debugging step.

If this attempt fails, look at the permission flags on the sensor of the */root/.ssh* and the */etc/ssh* directories. The directories themselves should be owned by root with the following permissions:

```
ls -ld /etc/ssh /root/.ssh

drwxr-xr-x    2 root     root         4096 Oct 18 10:37 /etc/ssh
drwx------    2 root     root         4096 May  2  2002 /root/.ssh

cd /root/.ssh
ls -l

-rw-------    1 root     root         2363 Nov 28  2001 authorized_keys
-rw-------    1 root     root          736 Jan  2  2001 id_dsa
-rw-r--r--    1 root     root          610 Nov 20  2001 id_dsa.pub
-rw-------    1 root     root          535 Nov 20  2001 id_rsa1
-rw-r--r--    1 root     root          339 Nov 20  2001 id_rsa1.pub
-rw-------    1 root     root          951 Nov 20  2001 id_rsa2
-rw-r--r--    1 root     root          230 Nov 20  2001 id_rsa2.pub


cd /etc/ssh
ls -l

-rw-------    1 root     root        88039 Oct 18 10:29 moduli
-rw-r--r--    1 root     root         1140 Oct 18 10:29 ssh_config
-rw-------    1 root     root          668 Oct 10  2000 ssh_host_dsa_key
-rw-r--r--    1 root     root          604 Oct 10  2000 ssh_host_dsa_key.pub
-rw-------    1 root     root          529 Oct 10  2000 ssh_host_key
-rw-r--r--    1 root     root          333 Oct 10  2000 ssh_host_key.pub
-rw-------    1 root     root          887 Feb 20  2001 ssh_host_rsa_key
-rw-r--r--    1 root     root          210 Feb 20  2001 ssh_host_rsa_key.pub
```

```
-rw-r--r--    1 root     root        14901 Jun 27 07:57 ssh_known_hosts
-rw-------    1 root     root         2444 Oct 18 10:29 sshd_config
```

After the permissions on the files and directories have been reset, try step 2 again.

3. The final recommendation for debugging the analyzer to sensor connection is to go to the sensor and type `/etc/rc.d/init.d/sshd stop`. Then try `/usr/sbin/sshd -d`, which will start the *sshd* daemon in debug mode. By attempting the connection from the analyzer of step 2 above, the *sshd* daemon on the sensor will write debugging messages similar to those of the `ssh -v` shown in step 2, but from the perspective of the other end of the connection. This information should help debug connections that fail.

4. If all else fails, dash to the closest bookstore and purchase a copy of [2].

5.2.6.6 Install and Configure NMAP.

One of the tools available in the SHADOW tool window allows the analyst to run *nmap* to obtain some reconnaissance about an IP address or system name of interest. For a local system, *nmap* can ascertain what ports are open to help identify why that particular system is a target of outside probes. It is not advisable to use *nmap* on machines outside the local purview. *Nmap* is a valuable tool, but it can be disruptive to some systems. Use it with great care. *Nmap* is furnished with Red Hat Version 8.0 and later, so it can be installed from the distribution CDs. See section 5.2.7.5 for configuring the *nmap.cgi* script.

5.2.6.7 Configure SUDO.

SUDO (superuser do) is a command furnished with Red Hat Linux. This command allows an ordinary user to execute a particular set of commands with root privileges. It provides extensive logging of its use and the command it attempts to execute. Since the user “apache” is the owner of the *httpd* process, when someone clicks the “*NMAP”* button on their SHADOW toolbar, the *nmap.cgi* script is running as apache. In order to utilize all the power of *nmap*, root privileges are required. *Sudo* gives shadow root privileges to execute the *nmap* command. Here is the configuration of the `/etc/sudoers` file:

```
 1  # sudoers file.
 2  #
 3  # This file MUST be edited with the 'visudo' command as root.
 4  #
 5  # See the sudoers man page for the details on how to write a sudoers file.
 6  #
 7
 8  # Host alias specification
 9
10  Host_Alias      SHADOW = analyzer1.righteousdudes.com, \
11                           analyzer2.righteousdudes.com, \
12                           analyzer3.righteousdudes.com
13
14  # User alias specification
15
16  # Cmnd alias specification
17
18  Cmnd_Alias      NMAP = /usr/bin/nmap, /usr/local/bin/nmap
19
20  # User privilege specification
21
22  root          ALL=(ALL)                                 ALL
```

```
23  apache  SHADOW=NOPASSWD:                NMAP
```

Lines 10-12 define an alias for three analyzer hosts in the *righteousdudes.com* domain. Throughout the rest of this file, the alias “SHADOW” will stand for the names of the three analyzers *analyzer1.righteousdudes.com*, *analyzer2.righteousdudes.com*, and *analyzer3.righteousdudes.com*. Likewise, Line 18 defines an alias, “NMAP” for the commands */usr/bin/nmap* or */usr/local/bin/nmap*. The last line of the file says that the user “apache”, who is the owner of the web httpd processes, has permission to execute any of the comands in the alias “NMAP” on any of the machines in the alias “SHADOW” without providing a password. *Sudo* provides a lot more capability than is needed for SHADOW, see the man page and [3]. To prevent just anyone from using *nmap,* Apache must be configured to require authentication for users who click on the NMAP key, see section 5.2.7.5.

5.2.7 Configure SHADOW Analyzer Software.

To be useful for a particular site, the SHADOW software needs configuration to adapt it to the network and system environment into which it was installed. This version of SHADOW attempts to overcome an often-heard criticism about difficulty of installation by simplifying the configuration process. One global analyzer file is used to define locations of commands and master directories. Other sensor specific files define subdirectories and sensor specific parameters.

5.2.7.1 Configure Global Analyzer Parameters.

New to SHADOW Version 1.8 is a system wide configuration file. An example is furnished in the distribution tarball in */usr/local/SHADOW/etc/shadow.conf*:

```
 1        #  SHADOW master configuration file. It is used by the SHADOW scripts
 2        #  to orient themselves.
 3        #
 4        #  /etc/shadow.conf  - SHADOW Version 1.8
 5        #                      Last changed 13 Dec 2002
 6        #
 7        #  NOTE: this is Perl code which is read by the SHADOW scripts
 8        #  as they are called into execution. As such, they must be in valid Perl
 9        #  syntax. You should modify this file to match your installation, but be
10        #  sure your modifications don't cause Perl errors, i.e. the quotes,
11        #  semicolons, equals signs, and dollar signs are required.
12        #
13        our $SHADOW_PATH = "/usr/local/SHADOW";
14        #
15        # The paths to the SHADOW sub-commands are here. Modify these if you want to
16        # specify a different path.
17        #
18        our $SSH_CMD = "/usr/bin/ssh";
19        our $SCP_CMD = "/usr/bin/scp";
20        our $TCPDUMP_CMD = "/usr/sbin/tcpdump";
21        our $GUNZIP_CMD = "/bin/gunzip";
22        our $BUNZIP2_CMD = "/usr/bin/bunzip2";
23        our $GZIP_CMD = "/bin/gzip";
24        our $BZIP2_CMD = "/usr/bin/bzip2";
25        our $LOOKUP_CMD = "/usr/bin/host";
26        our $WHOIS_CMD = "/usr/bin/whois";
27        our $AT_CMD = "/usr/bin/at";
28        our $NMAP_CMD = "/usr/bin/nmap";
29        our $SUDO_CMD = "/usr/bin/sudo";
30        our $DF_CMD = "/bin/df";
31        our $CAT_CMD = "/bin/cat";
32        #
33        # Paths to various SHADOW components.
```

```
34          #
35          our $SHADOW_SENSOR_PATH =  "$SHADOW_PATH/sensor";
36          our $SHADOW_SITE_PATH = "$SHADOW_PATH/sites";
37          our $SHADOW_FILTER_PATH = "$SHADOW_PATH/filters";
38          our $SHADOW_CGI_PATH = "$SHADOW_PATH/httpd/cgi-bin";
39          #
40          our $SHADOW_RAW_DATA_PATH = "/usr/local/SHADOW/data";
41          our $SHADOW_WEB_DATA_PATH = "/home/shadow/html";
42          our $SHADOW_REL_WEB_PAGES_ROOT = "/tcpdump_results";
43          our $SHADOW_WEB_PAGES_PATH = "${SHADOW_WEB_DATA_PATH}${SHADOW_REL_WEB_PAGES_ROOT}";
44          #
45          ##############################################################################
46          #
47          # The following parameters are used by tools.cgi, search.cgi, and
48          # compose_IR.cgi to identify the set of sensors (and therefore the
49          # subdirectories into which the data is stored) which
50          # provide data, and the label to be placed in the scrolling lists
51          # of each of the CGI scripts to reference the particular sensor's data.
52          #
53          #
54          our %SENSOR_LABELS = (
55                                "Sensor1" => "Outside Perimeter",
56                                "Sensor2" => "Betw. Perim & Firewall",
57                                "Sensor3" => "Inside Firewall"  ,
58                               );
59          #
60          our @SENSORS = ("Sensor1", "Sensor2", "Sensor3");
61          #
62          our $DEFAULT_SENSOR = "Sensor3";
63          #
64          ##############################################################################
65          #
66          # The following parameters are used by compose_IR.cgi to build incident
67          # reports. Modify them to customize compose_IR.cgi for your site.
68          #
69          our @IR_HEADING = (
70                            "Righteous Dudes Industries - Network Security Division",
71                            "Intrusion Detection Incident Report",
72                            "Phone: 000-555-1212"
73                            );
74          our $IR_LABEL = "RDI Intrusion Detection Incident Report No.: ";
75          our $IR_TGT_LABEL = "Righteous Dudes Industries Incorporated - Podunk HQ";
76          our $IR_SEQNO_PREFIX = "RDI-IDR";
77          our $MAIL_SUB_PREFIX = "";
78          our $MAIL_SUB_SUFFIX = "\@ righteousdudes.com";
79          our $MAIL_SENDER = "From: The Righteous Dudes Team <shadow\@righteousdudes.com>";
80          #
81          our $SHADOW_IR_SEQNO_FILE = "$SHADOW_PATH/Reports/IR_seq";
82          our $SHADOW_IR_DATA_FILE = "$SHADOW_PATH/Reports/Incident-Reports";
83          #
84          our @INCIDENT_TYPES = (
85                                 'Denial of Service Attempt',
86                                 'IMAP Connection Attempt',
87                                 'Remote Login Attempt',
88                                 'RESET Scan',
89                                 'SYN/RST Scan',
90                                 'FTP Scan',
91                                 'Port Scan',
92                                 'POP3 Scan',
93                                 'SNMP Probe/Scan',
94                                 'RPC/Portmap Connection Attempt',
95                                 'DNS Zone Transfer Attempt',
96                                 'Single System Connection Attempt',
97                                 'SOCKS Exploit',
98                                 'Informational Report',
99                                 'Multiple Target/Port Scan or Connection Attempts',
100                                'Network Mapping Attempt',
101                                'Unknown Probe type',
```

```
                            'ICMP Scan',
                            'DNS Scan',
                            'NETBIOS Scan',
                            'Unknown UDP Event',
                            'Unknown ICMP Event',
                            'Unknown TCP Event',
                            'Possible IP Spoofing Event',
                            );
        #
        our $DEFAULT_INCI_TYPE = 'Informational Report';
        #
        # Define the list of email Incident Report recipients for the raw data:
        #
        our $RAW_RECIPIENTS = "honcho@righteousdudes.com";
        #
        # Define the list of email Incident Report recipients for the obfuscated data:
        #
        our $OBF_RECIPIENTS = "handler@subsidiary.com";
        #
        ##############################################################################
        #
        # The name of the system which serves all the SHADOW data files.
        # If undefined, fetchem.pl will assume it is to fetch the sensor
        # files from the machine on which it is executing.
        #
        our $SHADOW_FILE_SERVER = "raid01.righteousdudes.com";
        #
        ##############################################################################
        #
        # For the obfuscate.pl script and the statistics scripts, define an array of
        # local addresses with the CIDR notation the netmask, e.g.
        #
        # 172.16.0.0/16 where 16 is the number of network mask bits.
        #
        # For the obfuscate.pl script define an array of addresses into which to
        # obfuscate the local addresses into. There must be one obfuscation address
        # for each local address.
        #
        our @LOCAL_IP = (
                         "172.16.0.0/16", "172.31.66.0/24"
                        );
        our @OBF_IP = (
                       "192.168.0.0", "192.168.245.0"
                      );
        #
        ##############################################################################
        #
        # Part of the obfuscation process is to take real domain addresses and
        # email addresses that may be part of the data and obfuscate them as
        # well as the IP addresses. Define those domain names that you want
        # obfuscated here:
        #
        our @LOCAL_DOMAIN = (
                             "righteousdudes.com",
                             "goodfellows.com"
                             );
        #
        ##############################################################################
        #
        # The statistics scripts require a definition of how many entries to place
        # on each page of top talkers. It may be reset by adjusting the values below.
        #
        our $LOCAL_ADDRESSES_NUMBER = 50;
        our $REMOTE_ADDRESSES_NUMBER = 50;
        #
        ##############################################################################
        1;
```

This is a Perl header file, (normally a ".ph") file that is read by each of the SHADOW analyzer scripts to access globally defined parameters. The file, or a link to it, is expected in the location */etc/shadow.conf*. This file should be tailored to reflect the values for parameters that SHADOW scripts will use. Note that the actual file does not include the line numbers. The numbers are added for this report to assist in configuring the parameters. Line 13 defines where SHADOW should look for the unpacked tarball; where the scripts are stored. Lines 18-31 define the locations of the system commands that SHADOW will use in its scripts. If these commands are located in different places, insert the correct paths here. For lines 35-43, various subdirectories of the SHADOW system are defined, the paths where SHADOW will look for files or attempt to create them. Tailor them as desired.

Lines 54-62 define the names, labels, and the default sensors for the sensors served by this analyzer as will be displayed in the SHADOW tool window. Lines 69-119 are parameters used by *compose_IR.cgi*, the script that builds an incident report. Tailor these lines as to what information should appear on an incident report, including organization name, the prefix for the incident report number, headings for the form, a list of incident types to choose when filling out the report, the default incident type, and a list of persons to receive the reports both in raw and obfuscated forms. Each of these parameters should be customized as desired.

Line 127 names the machine in a possible SHADOW analyzer cluster that holds the *tcpdump* data files from the sensors. If this parameter is defined, the machine named here will be tasked to do the file fetching. If it is not defined, the machine running the *fetchem.pl* script will directly fetch the data. Since the "file server" will host the data files anyway, it makes more sense to have it fetch the data directly.

Lines 140-145 are used by the scripts *obfuscate.pl* and *statistics.ph* to define the concepts of "local" addresses. Using the examples as a pattern, create entries for each IP address range that is to be considered "local." Include in lines 155-156 the domain names should be considered "local." The *obfuscate.pl* script will examine any text file and change all "local" fully qualified domain addresses, including email, and IP addresses to obfuscated addresses that can be shared with other organizations. The statistics scripts use the concept of "local" to categorize traffic in statistical summaries according to the direction of travel of the packets, e.g. incoming, outgoing, etc. The values defined in lines 164 and 165 represent the number of "local" and "remote" sites to be included in the web pages generated by the statistical scripts. Change them as needed.

By putting so many user defined parameters into a single configuration file, the process of installing SHADOW is shortened and simplified. Previously, each of the individual scripts required localization in an error prone configuration process.

> [***Note:***]The */etc/shadow.conf* file is **NOT** used by the sensor scripts, for this version of SHADOW. For sensors, this file need not be built. See section 5.1.8.

5.2.7.2 Configure the Individual Sensor (Site) Configuration Files.

In the subdirectory */usr/local/SHADOW/sites,* an entry needs to be configured for each sensor to which the analyzer is going to regularly connect. In previous versions of SHADOW, a "site" was considered a sensor, the source of the *tcpdump* data. However, it is certainly possible to look at data from more than one viewpoint. One set of *tcpdump* filters that concentrate on incoming traffic would give the viewpoint of a conventional intrusion detection system. The same data could be looked at from a different viewpoint by defining another filter set to concentrate on outgoing traffic, in order to see what an

organization's personnel are accessing outside their local network. At the time of this release, the possibilities of creating multiple “viewpoints” from one set of data are being explored. Not all the SHADOW infrastructure has been developed yet to seamlessly permit this extension of the “site” concept.

In the meantime, configure the entries in this file to reflect information particular to each sensor. The “Site” that is used in the name of the file and the subdirectory into which the *tcpdump* data and web pages are stored is an identifier selected to connote the source or viewpoint of the data, such as perhaps, the sensor's name. The sample Site1.ph furnished in the SHADOW distribution is included:

```
 1  #  Variables needed by the analyzer scripts. Tailor this file
 2  #  to define the paths for different sensor sites.
 3  #
 4  # Site.ph            - SHADOW Version 1.8
 5  #                      Last changed 27 Nov 2002
 6  #
 7  # The '$SITE" name is a name used to identify a sensor subdirectory. This name
 8  # will be used to create subdirectories under the analyzer raw data and web
 9  # pages directories as specified by the parameters $SHADOW_RAW_DATA_PATH and
10  # $SHADOW_WEB_DATA_PATH in /etc/shadow.conf. The $SITE_LABEL is the label that
11  # will be placed on the scrolling lists in the tools.cgi web page and the
12  # search.cgi tool.
13  #
14  $SITE="Site1";
15  $SITE_LABEL = "Outside Network Perimeter";
16  #
17  #
18  # Put here the name of the machine on which the SHADOW sensor software is
19  # located. The analyzer fetches the raw data from the sensor.
20  #
21  $SENSOR="sensor01.righteousdudes.com";
22  #
23  # Put here the name of the machine which runs Apache to serve up your SHADOW
24  # generated web pages.
25  #
26  $WEB_SERVER="www.righteousdudes.com";
27  #
28  # Change the following line to reflect the directory on your sensor in which
29  # the raw sensor data is stored.
30  #
31  $SENSOR_DIR="/LOG";
32  #
33  # The following line reflects the directory on your analyzer machine
34  # into which the raw sensor data is fetched. The variable $SHADOW_RAW_DATA_PATH
35  # is defined in /etc/shadow.conf.
36  #
37  $ANALYZER_DIR="$SHADOW_RAW_DATA_PATH/$SITE";
38  #
39  # The following line reflects the directory on your analyzer machine where
40  # SHADOW will create the web pages which hold the filtered data. The variable
41  # $SHADOW_WEB_PAGES_PATH is defined in /etc/shadow.conf.
42  #
43  $OUTPUT_WEB_DIR="$SHADOW_WEB_PAGES_PATH/$SITE";
44  #
45  # The following variable reflects the relative path from the DocumentRoot
46  # variable defined in the Apache configuration files to the actual html files.
47  # The variable $SHADOW_REL_WEB_PAGES_ROOT is defined in
48  # /etc/shadow.conf.
49  #
50  $URL_OUTPUT_DIR="$SHADOW_REL_WEB_PAGES_ROOT/$SITE";
51  #
52  # The following line reflects where the filters for the site $SITE are
53  # stored. The variable $SHADOW_FILTER_PATH is defined in /etc/shadow.conf.
54  #
```

```
55  $FILTER_DIR="$SHADOW_FILTER_PATH/$SITE";
56  #
57  # The SCAN_THRESHOLD is the number of different IPs that a "foreign" machine
58  # can contact before SHADOW lists that foreign machine as a possible scanner.
59  # Change it to reflect your preference.
60  #
61  $SCAN_THRESHHOLD = "100";
62  #
63  # Set the following variable to the number of days you want to keep the
64  # raw data files on your sensor's disks before the cleanup.pl script removes
65  # them. It depends on the sizes of your files, the amount of sensor disk space,
66  # and your taste.
67  #
68  $CLEAN_TIME = 2;
69  #
70  ##############################################################################
71  #
72  # For the obfuscate.pl script and the statistics scripts, define an array of
73  # local addresses with the CIDR notation, e.g.
74  #
75  # 172.16.0.0/16 where 16 is the number of network mask bits.
76  #
77  # Note: if this section of code is placed in a "site.ph" file, it will
78  # override the definitions in the /etc/shadow.conf file. In that way,
79  # what is "local" and "external" can be dependent on the data viewport,
80  # i.e. the sensor, the filter set, and the local definition.
81  #
82  # For obfuscate.pl, only the definitions of @LOCAL_IP, @OBF_IP, and
83  # @LOCAL_DOMAIN contained in /etc/shadow.conf will be used, since
84  # obfuscate.pl is not "site" dependent.
85  #
86  if (defined @LOCAL_IP) {
87     @LOCAL_IP = ();
88     undef @LOCAL_IP;
89  }
90  our @LOCAL_IP = (
91                   "172.16.0.0/16", "172.31.66.0/24"
92                  );
93  #
94  # For the obfuscate.pl script define an array of addresses into which to
95  # obfuscate each of the local addreses defined above. There must be one
96  # obfuscation address for each local address.
97  #
98  if (defined @OBF_IP) {
99     @OBF_IP = ();
100    undef @OBF_IP;
101 }
102 our @OBF_IP = (
103                "192.168.0.0", "192.168.245.0"
104
105               );
106 #
107 # Part of the obfuscation process is to take real domain addresses and
108 # email addresses that may be part of the data and obfuscate them as
109 # well as the IP addresses. Define those domain names that you want
110 # obfuscated here:
111 #
112 if (defined @LOCAL_DOMAIN) {
113    @LOCAL_DOMAIN = ();
114    undef @LOCAL_DOMAIN;
115 }
116 our @LOCAL_DOMAIN = (
117                      "righteousdudes.com",
118                      "goodfellows.com"
119                      );
120 #
121 #
122 ##############################################################################
```

```
123  1;
```

The line numbers in the listing of Site.ph are there only for clarity in this document; the line numbers are not present in the actual file furnished with the SHADOW package. Lines 14 through 55 are configuration parameters to identify the name of the "site", its label on the web pages, the name of the sensor from which this "site's" data is to be retrieved, and the directories where "site"-specific files are located. The predefined parameters such as $SHADOW_RAW_DATA_PATH are defined in the global */etc/shadow.conf* file. Additional parameters are defined in lines 61-68 to identify the scan threshold for the *find_scan.pl* script, and the number of days of raw data to leave on the sensor. Lines 70 through 120 identify the concept of "local" for IP addresses and domains exactly as was described in section 5.2.7.1 for the */etc/shadow.conf* configuration file. The definition of local may differ from sensor to sensor, depending on the network architecture. These lines may be eliminated if local IPs and domain names are independent of sensor location. Otherwise, the definition of local in this file will supercede that defined in */etc/shadow.conf*.

One "Site.ph" file must be created and configured for each sensor from which the analyzer will obtain data. Once the system is fully operational, the parameters in these sensor specific files can be tweaked as required.

5.2.7.3 Create the Required Directories.

When the SHADOW tarball was unpacked, it created the necessary subdirectories under the */usr/local/SHADOW* directory, which by default is defined as $SHADOW_PATH in the */etc/shadow.conf* configuration file. For the production analyzer, other directories need to be created and ownership given to shadow:

```
1. mkdir -p $SHADOW_RAW_DATA_PATH
    where $SHADOW_RAW_DATA_PATH is replaced by the value of $SHADOW_RAW_DATA_PATH from /etc/shadow.conf as the analyzer
    directory which will hold the raw data for each of your sites.
2. mkdir -p $SHADOW_WEB_PAGES_PATH
    where $SHADOW_WEB_PAGES_PATH is replaced by the value of $SHADOW_WEB_PAGES_PATH from /etc/shadow.conf as the
    analyzer directory which will hold the web pages.
3. mkdir -p $SHADOW_IR_DATA_FILE
    where $SHADOW_IR_DATA_FILE is replaced by the value of $SHADOW_IR_DATA_FILE from /etc/shadow.conf as the directory
    on the analyzer where completed incident reports will be stored.

For each "site" corresponding to each sensor configured with a "Site.ph" file in section 5.2.7.2, the following
directories need to be created:

4. mkdir –p $FILTER_DIR
    where $FILTER_DIR is replaced by the value of $FILTER_DIR from $SHADOW_FILTER_PATH/{SITE} as the analyzer directory
    in which the tcpdump filters will be stored for the site identified by {SITE}.
5. mkdir –p $ANALYZER_DIR
    where $ANALYZER_DIR is replaced by the value of $ANALYZER_DIR from $SHADOW_RAW_DATA_PATH/{SITE}, and {SITE} is
    replaced by the identifier of the site or sensor whose raw data will be placed there.
6. mkdir –p $OUTPUT_WEB_DIR
    where $OUTPUT_WEB_DIR is replaced by the value of $OUTPUT_WEB_DIR from $SHADOW_RAW_DATA_PATH/{SITE} as the analyzer
    directory which will hold the web pages for the site identified by {SITE}.

After all the directories have been created, set their ownership to the shadow user:

7. chown -R shadow:shadow $SHADOW_PATH
    where  $SHADOW_PATH is replaced by the value of $SHADOW_PATH from /etc/shadow.conf as the analyzer directory which
    will hold the web pages. By default, $SHADOW_PATH has the value "/usr/local/SHADOW."
8. chown –R shadow:shadow $SHADOW_RAW_DATA_PATH
```

```
9. chown -R shadow:shadow $SHADOW_WEB_PAGES_PATH
```

5.2.7.4 Configure the *Tcpdump* Filters.

In the */usr/local/SHADOW/filters* directory are seven files: *filter.getall.doc, goodhost.filter.doc, icmp.filter.doc, ip.filter.doc, tcp.filter.doc, and udp.filter.doc*. These are the sample filter files, commented to explain the function of each line. Each file should be edited to replace the generic IP addresses with site-specific IP addresses. There is one file for each protocol in the IP protocol suite. The *goodhost.filter.doc* covers special purpose machines in a site, e.g. a mail server, or web servers. The *filter.getall* is used by the *find_scan.pl* script to identify those IP addresses considered to be "inside" a specific site. After editing each file, strip the comments out by using the *comment_strip* script, for example:

```
/usr/local/SHADOW/comment_strip ip.filter.doc > /usr/local/SHADOW/filters/{SITE}/ip.filter
/usr/local/SHADOW/comment_strip icmp.filter.doc > /usr/local/SHADOW/filters/{SITE}/icmp.filter
/usr/local/SHADOW/comment_strip tcp.filter.doc > /usr/local/SHADOW/filters/{SITE}/tcp.filter
/usr/local/SHADOW/comment_strip udp.filter.doc > /usr/local/SHADOW/filters/{SITE}/udp.filter
/usr/local/SHADOW/comment_strip goodhost.filter.doc > /usr/local/SHADOW/filters/{SITE}/goodhost.filter
/usr/local/SHADOW/comment_strip filter.getall.doc > /usr/local/SHADOW/filters/{SITE}/filter.getall
```

A set of filters must be configured for each of the "sites" for which the analyzer will construct a web page. The filters may differ based on the type of traffic to be displayed on the web page. For example, the filters for a sensor outside a site's firewall may be configured to watch for incoming *telnet* connections, while those for another sensor inside the firewall may be configured to ignore them. After each filter is created, verify that the *tcpdump* command will parse it by (running as root):

```
tcpdump -i eth0 -n -F /usr/local/SHADOW/filters/{SITE}/XXX.filter
```

If that command does not give a "parse error," then the filter is syntactically correct. Whether the web page displays the items intended is another matter. The filter files provided in the SHADOW distribution are intended to serve as templates, subject to local modifications. One of the most important tasks a SHADOW analyst must do is to periodically modify the filters to reflect the traffic that is "normal" to his site in order to display the abnormal traffic on the web page . Books have been written to provide assistance in configuring the *tcpdump* filters. References [4] and [5] are good ones. Go to http://www.sans.org for information on training in Intrusion Detection and assistance in configuration of "signatures."

5.2.7.5 Configure the CGI Scripts.

In the directory */usr/local/SHADOW/httpd/cgi-bin*, are the CGI (Common Gateway Interface) scripts used by the SHADOW system to generate web pages for its tools. Previous versions of SHADOW required customization of each script in a tedious, error-prone process. With SHADOW Version 1.8 and the */etc/shadow.conf* configuration file, customization has been simplified. Each of the scripts accesses needed parameters from */etc/shadow.conf* .

The *nmap* function of the bottom button of the tools window needs some additional configuration. Sections 5.2.6.6, and 5.2.6.7 alluded to using *nmap* from the SHADOW tool window. *Sudo* was installed to give the web browser account (apache) permission to use *nmap* with root privileges. But it is desirable

to restrict *nmap* usage even further; not everyone who can access and examine SHADOW data should be allowed to run *nmap* as root. That restriction can be enforced by placing a file named *.htaccess* in the directory where the *nmap.cgi* script is located, (*/usr/local/SHADOW/httpd/cgi-bin/privileged* by default). Be aware as well that *nmapping* an address outside a site's domain may also be restricted by official policy and firewall rules. Here are the contents of the *.htaccess* file provided in the SHADOW tarball:

```
AuthType Basic
AuthName "Privileged SHADOW Users"
AuthUserFile /usr/local/SHADOW/httpd/cgi-bin/privileged/nmap_pwd
Satisfy any
require valid-user
order deny,allow
allow from 172.16.47
deny from all
```

This file accomplishes the following: It allows anyone from the "privileged" subnet, 172.16.47, to execute any command in this directory, i.e. *nmap.cgi*. For those not on the privileged subnet, if the user has a valid name/password combination in the file *nmap_pwd* in the same directory, he can run the *nmap.cgi* script as well. All other users who attempt to execute the script will be denied. To create user/password pairs in the file *nmap_pwd*, use the *htpasswd* command that is furnished as part of the Apache distribution.

```
htpasswd -c /usr/local/SHADOW/httpd/cgi-bin/privileged/nmap_pwd ImaUser
```

This command will create the *nmap_pwd* file and add the user "ImaUser" to it, prompting for a password twice. Use the command without the "-c" option to add additional users. Please see Ref [3] for more details about how to set up Apache authentication for individual subdirectories.

5.2.8 Test the SHADOW Scripts.

At this point, the SHADOW system should be set up. Now, the task is to test the SHADOW scripts to verify that everything works as intended. After one of the sensors has collected at least an hour's worth of data, test the SHADOW analyzer by running:

```
/usr/local/SHADOW/fetchem.pl -l Site1 -d YYYYMMDDHH -debug

where YYYY=year, MM=2-digit month, DD=2-digit day of the month, and HH=hour of the day of the collected data.
```

This command will run the *fetchem.pl* script which will:

1. Create a debug file /tmp/fetchem.log
2. Load the site (sensor) specific information from the header file */usr/local/SHADOW/sites/Site1.ph.*
3. Create the subdirectories on the analyzer to hold the *tcpdump* data file and the output html file, if necessary.
4. Use SSH to see if the necessary *tcpdump* file exists on the sensor and abort if it doesn't.
5. Use *scp* to copy the compressed data file from the sensor to its place on the analyzer.
6. Create an output HTML file and write the necessary header information to it.
7. Start uncompressing the data file, and feed the data to as many *tcpdump* processes as necessary for each of the filters in the sensor filter directory, plus one more for the *find_scan.pl* script.
8. Collect all the output text files from the execution of the *tcpdump* processes into a single text file.
9. Sort the concatenated text file by IP address, and convert the IP addresses to domain names where possible.
10. Copy the sorted and resolved text file to the HTML file.
11. Add the information from the *find_scan.pl* script to the HTML file, then add the navigation bar to the end along with the necessary closing HTML statements.
12. Remove all the temporary text files created by this run and exit.

With the "-debug" flag set on the *fetchem.pl* script call, logging information will be written to /tmp/fetchem.log at many places in the script. By examining each line in that log file, the corresponding place in the *fetchem.pl* script where the message was written can be found. If the script aborts for some reason, the lines in the log will indicate the area of the script that failed. When the process completes normally, there will be an HTML file created in:

/home/shadow/html/tcpdump_results/{SITE}/Mondd/YYYYMMDDHH.html.

or whever the *$OUTPUT_WEB_DIR* variable of the */usr/local/SHADOW/sites/Site1.ph* indicates.

5.2.9 Configure Crontab on the Analyzer.

The SHADOW release package contains a file */usr/local/SHADOW-1.8/analyzer_crontab.shadow* which contains:

```
 1  # DO NOT EDIT THIS FILE - edit the master and reinstall.
 2  # (/tmp/crontab.1761 installed on Wed Feb  6 13:13:40 2002)
 3  # (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
 4  #
 5  # Run fetchem to get SHADOW data files:
 6  #
 7  SHADOW_PATH=/usr/local/SHADOW
 8  #
 9  1 * * * * $SHADOW_PATH/fetchem.pl -l Site1
10  3 * * * * $SHADOW_PATH/fetchem.pl -l Site2 -debug
11  #
12  # Cleanup once per day.
13  #
14  15 1 * * * $SHADOW_PATH/cleanup.pl -l Site1
15  24 1 * * * $SHADOW_PATH/cleanup.pl -l Site2
16  #
17  # Collect statistics each night.
```

```
18  #
19  1 0 * * * $SHADOW_PATH/stats/do_daily_stats.pl -l Site1
20  1 3 * * * $SHADOW_PATH/stats/do_daily_stats.pl -l Site2
21
```

This file is an example crontab that should be run be run as the user "shadow" on the analyzer. Line 9 says that at 5 minutes past each hour, the script "`/usr/local/SHADOW/fetchem.pl -l Site1`" will be run. There should be a line like this with the appropriate sensor name substituted for "Site1." If there is more that one sensor, the crontab will need multiple lines, one for each. Line 14 says that the "`/usr/local/SHADOW/cleanup.pl -l Site1`" script will be run at 1:17 (am) each day. Line 19 says that the "`/usr/local/SHADOW/do_daily_stats.pl -l Site1`" script will be run at 0:19 (12:19am) each day.

The SHADOW release package contains a file */usr/local/SHADOW-1.8/analyzer_crontab.root* which contains:

```
 1  #               SHADOW Version: 1.8
 2  #               Last Changed:   15 Jan 2003
 3  #
 4  # Crontab for a SHADOW Analyzer. We need to insure that the system clock
 5  # is consistent between tha analyzer and the sensor.
 6  #
 7  # If this system is not on the internet, we need to reference a standard
 8  # time source.
 9  #
10  # These two entries must be done as root.
11  #
12  17 23 * * * /usr/bin/ntpdate time-a.nist.gov
13  18 23 * * * /sbin/hwclock --systohc
14  #
```

Line 12 uses the *ntpdate* program to synchronize time with the *time-a.nist.gov* time server at 23:17 (11:17 PM). Line 13 says that at 23:18 the hardware clock will be set to the system clock which was just synchronized to the NTP time source. As stated in section 5.1.9, the system clocks of the analyzer and its sensor should be kept relatively synchronized.

5.2.10 Protect the SHADOW Analyzer.

A site protected by a firewall as part of a well-defined security policy, implies that the SHADOW analyzer does not require as much protection as the sensor. Nevertheless, as part of a "security in depth" philosophy, the analyzer should be configured to detect unwanted activity and protect itself, rather than totally depending totally on an external firewall. Prudent security precautions are always appropriate. Like the sensor, the system should be stripped of all network services except those necessary for SHADOW functionality, e.g. Httpd, SSH, etc. On Red Hat Linux 7.x, this involves removing all services from */etc/xinetd.conf* and either removing all the files in the */etc/xinetd.d* directory, or adding the line "disable = yes" to each. The *tcp_wrappers* system is furnished with Red Hat Linux as an additional security mechanism that allows access control lists on the TCP/IP services to which the system permits connection. For more help, see the *hosts.allow* and *hosts.deny* man pages and section 5.1.10.

In the SHADOW release package, and included as Appendix C:, is an example *rc.iptables* file used to protect a typical SHADOW analyzer. The script will allow IP communications on only those other

services that may be necessary, such as DNS, httpd, https, SMTP, NNTP, etc. It will block and log all other attempts to connect to the analyzer.

Recommended resources for more general information about Linux security, are Refs. [6], [7], [8] and [9].

5.3 Put the SHADOW System into Production.

At this point, everything should be working. The sensor should be capturing packets and writing the *tcpdump* files. The analyzer should be fetching those files, running them through the filters and generating web pages. The tools should be functional. The easy part of SHADOW configuration is complete. Now comes the difficult part: learning what a site's network traffic really looks like and adjusting the filters to reflect that reality. As more familiarity with the network is acquired, the filters will undoubtedly need to be periodically changed to reflect both improved understanding of the network topology and the ever-changing risks from both outside and inside. The art of creating filters for different traffic patterns is beyond the scope of this SHADOW installation document. The best books on the topic are Ref [4] and Ref [5]. They will assist in building a foundation of understanding on which the SHADOW system and better intrusion detection expertise can be built. Stay in touch with the SANS Institute http://www.sans.org to follow what's happening in intrusion detection and to get detailed training.

Happy detecting.

Appendix A: SHADOW Sensor Kernel Configuration

This is a copy of the kernel configuration file for a SHADOW sensor (taken from a Ver. 2.4.19 kernel build). Only those configuration options selected are printed out; the rest are left out to shorten the length of the file.

```
#
# Automatically generated make config: don't edit
#
CONFIG_X86=y
CONFIG_ISA=y
CONFIG_UID16=y
#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
#
# Loadable module support
#
#
# Processor type and features
#
CONFIG_M486=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_CMPXCHG=y
CONFIG_X86_XADD=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_X86_L1_CACHE_SHIFT=4
CONFIG_X86_USE_STRING_486=y
CONFIG_X86_ALIGNMENT_16=y
CONFIG_X86_PPRO_FENCE=y
CONFIG_NOHIGHMEM=y
CONFIG_MTRR=y
#
# General setup
#
CONFIG_NET=y
CONFIG_PCI=y
CONFIG_PCI_GOANY=y
CONFIG_PCI_BIOS=y
CONFIG_PCI_DIRECT=y
CONFIG_PCI_NAMES=y
CONFIG_SYSVIPC=y
CONFIG_SYSCTL=y
CONFIG_KCORE_ELF=y
CONFIG_BINFMT_AOUT=y
CONFIG_BINFMT_ELF=y
CONFIG_BINFMT_MISC=y
#
# Memory Technology Devices (MTD)
#
#
# Parallel port support
#
#
# Plug and Play configuration
#
#
# Block devices
#
CONFIG_BLK_DEV_FD=y
```

```
#
# Multi-device support (RAID and LVM)
#
#
# Networking options
#
CONFIG_PACKET=y
CONFIG_NETFILTER=y
CONFIG_NETFILTER_DEBUG=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_SYN_COOKIES=y
#
#   IP: Netfilter Configuration
#
CONFIG_IP_NF_CONNTRACK=y
CONFIG_IP_NF_FTP=y
CONFIG_IP_NF_IPTABLES=y
CONFIG_IP_NF_MATCH_LIMIT=y
CONFIG_IP_NF_MATCH_MAC=y
CONFIG_IP_NF_MATCH_MARK=y
CONFIG_IP_NF_MATCH_MULTIPORT=y
CONFIG_IP_NF_MATCH_TOS=y
CONFIG_IP_NF_MATCH_STATE=y
CONFIG_IP_NF_FILTER=y
CONFIG_IP_NF_TARGET_REJECT=y
CONFIG_IP_NF_TARGET_LOG=y
#
#
#
#
# Appletalk devices
#
#
# QoS and/or fair queueing
#
#
# Network testing
#
#
# Telephony Support
#
#
# ATA/IDE/MFM/RLL support
#
CONFIG_IDE=y
#
# IDE, ATA and ATAPI Block devices
#
CONFIG_BLK_DEV_IDE=y
#
# Please see Documentation/ide.txt for help/info on IDE drives
#
CONFIG_BLK_DEV_IDEDISK=y
CONFIG_BLK_DEV_IDECD=y
#
# IDE chipset support/bugfixes
#
CONFIG_BLK_DEV_IDEPCI=y
CONFIG_IDEPCI_SHARE_IRQ=y
CONFIG_BLK_DEV_IDEDMA_PCI=y
CONFIG_IDEDMA_PCI_AUTO=y
CONFIG_BLK_DEV_IDEDMA=y
CONFIG_BLK_DEV_ADMA=y
CONFIG_IDEDMA_AUTO=y
#
# SCSI support
#
CONFIG_SCSI=y
#
```

```
# SCSI support type (disk, tape, CD-ROM)
#
CONFIG_BLK_DEV_SD=y
CONFIG_SD_EXTRA_DEVS=40

#
# Some SCSI devices (e.g. CD jukebox) support multiple LUNs
#
CONFIG_SCSI_CONSTANTS=y
#
# SCSI low-level drivers
#
CONFIG_SCSI_AIC7XXX=y
CONFIG_AIC7XXX_CMDS_PER_DEVICE=8
CONFIG_AIC7XXX_RESET_DELAY_MS=5000
#
# Fusion MPT device support
#
#
# IEEE 1394 (FireWire) support (EXPERIMENTAL)
#
#
# I2O device support
#
#
# Network device support
#
CONFIG_NETDEVICES=y
#
# ARCnet devices
#
#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
CONFIG_NET_VENDOR_3COM=y
CONFIG_VORTEX=y
CONFIG_NET_PCI=y
CONFIG_TULIP=y
#
# Ethernet (1000 Mbit)
#
#
# Wireless LAN (non-hamradio)
#
#
# Token Ring devices
#
#
# Wan interfaces
#
#
# Amateur Radio support
#
#
# IrDA (infrared) support
#
#
# ISDN subsystem
#
#
# Old CD-ROM drivers (not SCSI, not IDE)
#
#
# Input core support
#
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
#
# Character devices
```

```
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_SERIAL=y
CONFIG_UNIX98_PTYS=y
CONFIG_UNIX98_PTY_COUNT=256
#
# I2C support
#
#
# Mice
#
CONFIG_MOUSE=y
CONFIG_PSMOUSE=y
#
# Joysticks
#
#
# Input core support is needed for gameports
#
#
# Input core support is needed for joysticks
#
#
# Watchdog Cards
#
CONFIG_RTC=y
#
# Ftape, the floppy tape device driver
#
#
# Multimedia devices
#
#
# File systems
#
CONFIG_EXT3_FS=y
CONFIG_JBD=y
CONFIG_JBD_DEBUG=y
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
CONFIG_TMPFS=y
CONFIG_RAMFS=y
CONFIG_ISO9660_FS=y
CONFIG_JOLIET=y
CONFIG_PROC_FS=y
CONFIG_DEVPTS_FS=y
CONFIG_EXT2_FS=y
#
# Network File Systems
#
#
# Partition Types
#
CONFIG_MSDOS_PARTITION=y
CONFIG_NLS=y
#
# Native Language Support
#
CONFIG_NLS_DEFAULT="cp437"
CONFIG_NLS_CODEPAGE_437=y
CONFIG_NLS_ISO8859_1=y
#
# Console drivers
#
CONFIG_VGA_CONSOLE=y
CONFIG_VIDEO_SELECT=y
#
# Frame-buffer support
```

```
#
#
# Sound
#
#
# USB support
#
#
# Bluetooth support
#
#
# Kernel hacking
#
```

Appendix B: SHADOW Analyzer Kernel Configuration

This is a copy of the kernel configuration file for a SHADOW analyzer (taken from a Ver. 2.4.19 kernel build). Only those configuration options selected are printed out; the rest are left out to shorten the length of the file.

```
#
# Automatically generated make config: don't edit
#
CONFIG_X86=y
CONFIG_ISA=y
CONFIG_UID16=y
#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
#
# Loadable module support
#
CONFIG_MODULES=y
CONFIG_KMOD=y
#
# Processor type and features
#
CONFIG_M686=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_CMPXCHG=y
CONFIG_X86_XADD=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_X86_L1_CACHE_SHIFT=5
CONFIG_X86_TSC=y
CONFIG_X86_GOOD_APIC=y
CONFIG_X86_PGE=y
CONFIG_X86_USE_PPRO_CHECKSUM=y
CONFIG_X86_PPRO_FENCE=y
CONFIG_NOHIGHMEM=y
CONFIG_MTRR=y
#
# General setup
#
CONFIG_NET=y
CONFIG_PCI=y
CONFIG_PCI_GOANY=y
CONFIG_PCI_BIOS=y
CONFIG_PCI_DIRECT=y
CONFIG_PCI_NAMES=y
CONFIG_SYSVIPC=y
CONFIG_BSD_PROCESS_ACCT=y
CONFIG_SYSCTL=y
CONFIG_KCORE_ELF=y
CONFIG_BINFMT_AOUT=y
CONFIG_BINFMT_ELF=y
CONFIG_BINFMT_MISC=y
CONFIG_PM=y
#
# Memory Technology Devices (MTD)
#
#
# Parallel port support
#
#
# Plug and Play configuration
#
```

```
#
# Block devices
#
CONFIG_BLK_DEV_FD=y
CONFIG_BLK_DEV_LOOP=m
#
# Multi-device support (RAID and LVM)
#
#
# Networking options
#
CONFIG_PACKET=y
CONFIG_PACKET_MMAP=y
CONFIG_NETFILTER=y
CONFIG_NETFILTER_DEBUG=y
CONFIG_FILTER=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_SYN_COOKIES=y
#
#   IP: Netfilter Configuration
#
CONFIG_IP_NF_CONNTRACK=y
CONFIG_IP_NF_FTP=y
CONFIG_IP_NF_IPTABLES=y
CONFIG_IP_NF_MATCH_LIMIT=y
CONFIG_IP_NF_MATCH_MAC=y
CONFIG_IP_NF_MATCH_MARK=y
CONFIG_IP_NF_MATCH_MULTIPORT=y
CONFIG_IP_NF_MATCH_TOS=y
CONFIG_IP_NF_MATCH_STATE=y
CONFIG_IP_NF_FILTER=y
CONFIG_IP_NF_TARGET_REJECT=y
CONFIG_IP_NF_TARGET_LOG=y
#
#
#
#
# Appletalk devices
#
#
# QoS and/or fair queueing
#
#
# Network testing
#
#
# Telephony Support
#
#
# ATA/IDE/MFM/RLL support
#
CONFIG_IDE=y
#
# IDE, ATA and ATAPI Block devices
#
CONFIG_BLK_DEV_IDE=y
#
# Please see Documentation/ide.txt for help/info on IDE drives
#
CONFIG_BLK_DEV_IDEDISK=y
CONFIG_BLK_DEV_IDECD=y
CONFIG_BLK_DEV_IDESCSI=m
#
# IDE chipset support/bugfixes
#
CONFIG_BLK_DEV_IDEPCI=y
CONFIG_IDEPCI_SHARE_IRQ=y
CONFIG_BLK_DEV_IDEDMA_PCI=y
CONFIG_IDEDMA_PCI_AUTO=y
```

```
CONFIG_BLK_DEV_IDEDMA=y
CONFIG_BLK_DEV_ADMA=y
CONFIG_BLK_DEV_PIIX=y
CONFIG_PIIX_TUNING=y
CONFIG_IDEDMA_AUTO=y
CONFIG_BLK_DEV_IDE_MODES=y
#
# SCSI support
#
CONFIG_SCSI=y
#
# SCSI support type (disk, tape, CD-ROM)
#
CONFIG_BLK_DEV_SD=y
CONFIG_SD_EXTRA_DEVS=40
CONFIG_CHR_DEV_ST=m
CONFIG_BLK_DEV_SR=m
CONFIG_BLK_DEV_SR_VENDOR=y
CONFIG_SR_EXTRA_DEVS=2
CONFIG_CHR_DEV_SG=m
#
# Some SCSI devices (e.g. CD jukebox) support multiple LUNs
#
CONFIG_SCSI_DEBUG_QUEUES=y
CONFIG_SCSI_MULTI_LUN=y
CONFIG_SCSI_CONSTANTS=y
CONFIG_SCSI_LOGGING=y
#
# SCSI low-level drivers
#
CONFIG_SCSI_AIC7XXX=y
CONFIG_AIC7XXX_CMDS_PER_DEVICE=8
CONFIG_AIC7XXX_RESET_DELAY_MS=5000
#
# Fusion MPT device support
#
#
# IEEE 1394 (FireWire) support (EXPERIMENTAL)
#
#
# I2O device support
#
#
# Network device support
#
CONFIG_NETDEVICES=y
#
# ARCnet devices
#
#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
CONFIG_NET_VENDOR_3COM=y
CONFIG_VORTEX=y
#
# Ethernet (1000 Mbit)
#
#
# Wireless LAN (non-hamradio)
#
#
# Token Ring devices
#
#
# Wan interfaces
#
#
# Amateur Radio support
#
```

```
#
# IrDA (infrared) support
#
#
# ISDN subsystem
#
#
# Old CD-ROM drivers (not SCSI, not IDE)
#
#
# Input core support
#
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_SERIAL=y
CONFIG_UNIX98_PTYS=y
CONFIG_UNIX98_PTY_COUNT=256
#
# I2C support
#
#
# Mice
#
CONFIG_MOUSE=y
CONFIG_PSMOUSE=y
#
# Joysticks
#
#
# Input core support is needed for gameports
#
#
# Input core support is needed for joysticks
#
#
# Watchdog Cards
#
CONFIG_RTC=y
#
# Ftape, the floppy tape device driver
#
CONFIG_AGP=m
CONFIG_AGP_INTEL=y
CONFIG_DRM=y
#
# DRM 4.1 drivers
#
CONFIG_DRM_NEW=y
CONFIG_DRM_R128=m
#
# Multimedia devices
#
#
# File systems
#
CONFIG_EXT3_FS=y
CONFIG_JBD=y
CONFIG_JBD_DEBUG=y
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
CONFIG_TMPFS=y
CONFIG_RAMFS=y
CONFIG_ISO9660_FS=y
```

```
CONFIG_JOLIET=y
CONFIG_NTFS_FS=m
CONFIG_PROC_FS=y
CONFIG_DEVPTS_FS=y
CONFIG_EXT2_FS=y
#
# Network File Systems
#
CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
CONFIG_NFSD=y
CONFIG_NFSD_V3=y
CONFIG_SUNRPC=y
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_SMB_FS=y
#
# Partition Types
#
CONFIG_MSDOS_PARTITION=y
CONFIG_SMB_NLS=y
CONFIG_NLS=y
#
# Native Language Support
#
CONFIG_NLS_DEFAULT="cp437"
CONFIG_NLS_CODEPAGE_437=y
CONFIG_NLS_ISO8859_1=y
#
# Console drivers
#
CONFIG_VGA_CONSOLE=y
CONFIG_VIDEO_SELECT=y
#
# Frame-buffer support
#
CONFIG_FB=y
CONFIG_DUMMY_CONSOLE=y
CONFIG_VIDEO_SELECT=y
CONFIG_FB_ATY128=y
CONFIG_FBCON_ADVANCED=y
CONFIG_FBCON_MFB=y
CONFIG_FBCON_CFB2=y
CONFIG_FBCON_CFB4=y
CONFIG_FBCON_CFB8=y
CONFIG_FBCON_CFB16=y
CONFIG_FBCON_CFB24=y
CONFIG_FBCON_CFB32=y
CONFIG_FONT_8x8=y
CONFIG_FONT_8x16=y
#
# Sound
#
CONFIG_SOUND=m
CONFIG_SOUND_ES1371=m
#
# USB support
#
#
# Bluetooth support
#
#
# Kernel hacking
#
```

Appendix C: Protecting the SHADOW System

The following is an example iptables shell script that can be put into the /etc/rc.d/rc.local script on both the sensor and the analyzer to help protect them from unauthorized network access attempts. The script needs customization to define the set of ANALYZERS, SENSORS, TRUSTED, and UNTRUSTED IP addresses.

```
#!/bin/sh
#
#
# This script sets up iptables chains to protect a SHADOW sensor or analyzer.
#
# written by Bill Ralph <RalphWD@nswc.navy.mil>
#
# Last modified: 03 Dec 2002
#
# ------------------------------------------------------------------------
#
# Define some variables.
#
LOG_LEVEL="notice"
LOOPBACK="127.0.0.1"
LOOPBACK_IF="lo"
ACTIVE_IF="eth0"
#
# ------------------------------------------------------------------------
#
ANALYZERS="172.21.230.165 172.21.230.166 172.21.230.167 172.21.230.168
           172.21.230.169 172.21.230.170 172.21.230.171"
#
SENSORS="172.21.230.172 172.21.230.173 172.21.230.174 172.21.230.175"
#
UNTRUSTED="172.21.230.176 172.21.230.177 172.21.230.178 172.21.230.179
           172.21.230.184 172.21.230.185 172.21.240.186"
#
TRUSTED="172.21.230.164"
#
# ------------------------------------------------------------------------
#
# Source function library.
. /etc/rc.d/init.d/functions

if [ ! -f /etc/sysconfig/network ]
then
    exit 0
fi

. /etc/sysconfig/network

#
# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -x /sbin/ifconfig ] || exit 0

[ -f /etc/sysconfig/network-scripts/ifcfg-$ACTIVE_IF ] || exit 0

. /etc/sysconfig/network-scripts/ifcfg-$ACTIVE_IF

# Add the NETWORK and BROADCAST variables.

if [ -z $NETWORK ]
then
   eval `/bin/ipcalc -n $IPADDR $NETMASK`
```

```
fi
if [ -z $BROADCAST ]
then
   eval `/bin/ipcalc -b $IPADDR $NETMASK`
fi

#
# ------------------------------------------------------------------------
#
# First, let's determine whether we're a sensor or analyzer.
#
WHAT_I_AM=""
#
for IP in $SENSORS
do
   if [ "$IP" = "$IPADDR" ]
   then
      WHAT_I_AM="SENSOR"
      break
   fi
done
#
if [ "$WHAT_I_AM" != "SENSOR" ]
then
   for IP in $ANALYZERS
   do
      if [ "$IP" = "$IPADDR" ]
      then
         WHAT_I_AM="ANALYZER"
         break
      fi
   done
fi
if [ "$WHAT_I_AM" = "" ]
then
   echo "Improper script configuration.\n"
   exit 1
fi
#
# ------------------------------------------------------------------------
#
# Next, flush any current chains, then remove any user-defined chains.
#
 iptables -F
 iptables -X
#
# ------------------------------------------------------------------------
#
# Make the default policy of the input chain DROP
#
 iptables -P INPUT DROP
#
# ------------------------------------------------------------------------
#
# Allow unlimited traffic to and from ourselves.
#
 iptables -A INPUT  -i $LOOPBACK_IF -j ACCEPT
 iptables -A OUTPUT -o $LOOPBACK_IF  -j ACCEPT
 iptables -A INPUT  -i $ACTIVE_IF -s $IPADDR/32 -j ACCEPT
 iptables -A OUTPUT -o $ACTIVE_IF -s $IPADDR/32 -j ACCEPT
#
# ------------------------------------------------------------------------
#
# If we are a sensor, allow nothing to go out on our passive interface(s).
#
if [ "$WHAT_I_AM" = "SENSOR" ]
then
   iptables -A OUTPUT -o ! $ACTIVE_IF -j DROP
fi
#
```

```
# ------------------------------------------------------------------------
#
# Create a chain to log information about a connection, then drop it.
#
 iptables -N log_drop
 iptables -A log_drop -m limit --limit 3/day --limit-burst 1 \
                 -j LOG --log-level $LOG_LEVEL --log-prefix "Dropped: "
 iptables -A log_drop -j DROP
#
# ------------------------------------------------------------------------
#
# Create a chain to reject a packet with a icmp-host-prohibited packet.
# Severerly limit the bandwidth accepted.
#
 iptables -N log_rej
 iptables -A log_rej -m limit --limit 12/day --limit-burst 3 \
                -j LOG --log-level $LOG_LEVEL --log-prefix "Rejected: "
 iptables -A log_rej -j REJECT --reject-with icmp-host-prohibited
#
# ------------------------------------------------------------------------
#
# Create a chain to specifically accept IP traffic on established
# connections that we initiated. Log traffic on INVALID connections.
#
 iptables -N estab
 iptables -A estab --match state --state INVALID -j LOG \
                  --log-prefix "INVALID conn:"
 iptables -A estab --match state --state INVALID -j DROP
 iptables -A estab --match state --state ESTABLISHED,RELATED -j ACCEPT
#
# ------------------------------------------------------------------------
#
# Create two chains to explicitly deny access to those labelled "untrusted"
# and explicitly grant access to those we trust.
#
# ------------------------------------------------------------------------
#
 iptables -N untrusted
 iptables -N trusted
#
# Unconditionally drop all packets with a broadcast destination.
# Unconditionally drop all SMB, NETBIOS, RIP traffic without a trace.
#
 iptables -A untrusted -d ${NETWORK}/32 -j DROP
 iptables -A untrusted -d ${BROADCAST}/32 -j DROP
 iptables -A untrusted -d 255.255.255.255/32 -j DROP
 iptables -A untrusted -p udp --dport 135:139 -j DROP
 iptables -A untrusted -p udp --dport 520 -j DROP
#
# ------------------------------------------------------------------------
#
#
 if [ "$UNTRUSTED" != "" ]
 then
    for IP in $UNTRUSTED
    do
       iptables -A untrusted -s $IP/32 -j DROP
    done
 fi
#
# ------------------------------------------------------------------------
#
# If I am a sensor, allow SSH and ICMP traffic from the analyzers
# and allow all from my TRUSTED administrators.
#
 if [ "$WHAT_I_AM" = "SENSOR" ]
 then
    for IP in $ANALYZERS
    do
       iptables -A trusted -p tcp -s $IP/32 --dport 22 -j ACCEPT
```

```
            iptables -A trusted -p icmp -s $IP/32 --icmp-type 8 -j ACCEPT
         done
         for IP in $TRUSTED
         do
            iptables -A trusted -s $IP/32 -j ACCEPT
         done
      #
      # ------------------------------------------------------------------------
      #
      # If I am an analyzer, allow intra-analyzer traffic.
      #
       elif [ "$WHAT_I_AM" = "ANALYZER" ]
       then
         for IP in $ANALYZERS
         do
            if [ "$IP" != "$IPADDR" ]
            then
               iptables -A trusted -s $IP/32 -j ACCEPT
            fi
         done
       fi
      #
      # ------------------------------------------------------------------------
      #
      # Create a chain to look at TCP packets. Examine the flags and limit the
      # weird ones we will accept. Limit the average matching rate to slow down
      # nmaps and such.
      #
       iptables -N tcp_pkts
       iptables -A tcp_pkts -p tcp --tcp-flags ALL FIN,URG,PSH -m limit \
                     --limit 6/hour -j LOG --log-level $LOG_LEVEL \
                     --log-prefix "NMAP-XMAS:"
       iptables -A tcp_pkts -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
       iptables -A tcp_pkts -p tcp --tcp-flags SYN,RST SYN,RST -m limit \
                     --limit 5/minute -j LOG --log-level $LOG_LEVEL \
                     --log-prefix "SYN/RST:"
       iptables -A tcp_pkts -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
       iptables -A tcp_pkts -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit \
                     --limit 5/minute -j LOG --log-level $LOG_LEVEL \
                     --log-prefix "SYN/FIN:"
       iptables -A tcp_pkts -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
      #
       if [ "$WHAT_I_AM" = "ANALYZER" ]
       then
      #
      # Accept https connections from anyone at righteousdudes.org. Deny all other
      # TCP packets on the well-known ports.
      #
          iptables -A tcp_pkts -p tcp -s 172.21.0.0/16 --dport 443 -j ACCEPT
      #
      # Accept SMTP connections from our mail relays.
      #
          iptables -A tcp_pkts -p tcp -s 172.21.200.187/32 --dport 25 -j ACCEPT
          iptables -A tcp_pkts -p tcp -s 172.21.200.188/32 --dport 25 -j ACCEPT
       fi
      #
      # Log all TCP SYN (connection) requests
      #
       iptables -A tcp_pkts -p tcp -i ! lo --syn -m limit --limit 6/h -j LOG \
                           --log-level $LOG_LEVEL --log-prefix "Dropped SYN:"
       iptables -A tcp_pkts -j DROP
      #
      # ------------------------------------------------------------------------
      #
      # Create a chain for UDP packets. Deny most, except NTP packets from
      # the timeserver and DNS packets from the nameservers.
      # Deny RIP packets from our router. Log all other udp packets.
      #
       iptables -N udp_pkts
       iptables -A udp_pkts -p udp -s 172.21.225.189/32 --sport 123 -j ACCEPT
```

```
 iptables -A udp_pkts -p udp -s 172.21.209.190/32 --sport 53 -j ACCEPT
 iptables -A udp_pkts -p udp -s 172.21.200.191/32 --sport 53 -j ACCEPT
 iptables -A udp_pkts -j log_drop
#
# ------------------------------------------------------------------------
#
# Apply the defined chains to the input chain.
#
 iptables -A INPUT -j estab
 iptables -A INPUT -j untrusted
 iptables -A INPUT -j trusted
 iptables -A INPUT -p tcp -j tcp_pkts
 iptables -A INPUT -p udp -j udp_pkts
```

Bibliography

[1] *The Official Red Hat Linux 7.1 x86 Installation Guide,* Red Hat, Inc., 2001

[2] *SSH The Secure Shell, The Definitive Guide,* Daniel J. Barrett and Richard E. Silverman, ISBN: 0-596-00011-1, O'Reilly and Associates, 2001

[3] *Unix System Administration Handbook, Third Edition,* Evi Nemeth, Garth Snyder, Scott Seebass, and Trent R. Hein, ISBN 0-13-020601-6, Prentice Hall PTR, 2001.

[4] *Network Intrusion Detection, An Analyst's Handbook, Second Edition,* Stephen Northcutt and Judy Novak, ISBN_0-7357-1008-2, New Riders Publishing, 2001.

[5] *Intrusion Signatures and Analysis,* Stephen Northcutt, Mark Cooper, Matt Fearnow, and Karen Frederick, ISBN 0-7357-1063-5, New Riders Publishing, 2001.

[6] *Maximum Linux Security,* Anonymous, ISBN 0-672-31670-6, SAMS Publishing, 2000.

[7] *Linux System Security,* Scott Mann and Ellen L. Mitchell, ISBN 0-13-015807-0, Prentice Hall PTR, 2000.

[8] *Real World Linux Security,* Bob Toxen, ISBN 0-13-028187-5, Prentice Hall PTR, 2001.

[9] *Linux Firewalls,* Robert L. Ziegler, ISBN 0-7357-0900-9, New Riders Publishing, 2000.

[10] *] Apache, The Definitive Guide, Second Edition,* Ben Laurie & Peter Laurie, ISBN 1-56592-528-9, O'Reilly and Associates, 1999.

[11] *Programming Perl, Third Edition,* Larry Wall, Tom Christiansen, and Jon Orwant, ISBN 0-596-00027-8, O'Reilly and Associates, 2000.

[12] *Official Guide to Programming with CGI.pm,* Lincoln Stein, ISBN: 0-471-24744-8, John Wiley & Sons, Inc., 1998.

[13] *TCP/IP Illustrated, Volume 1,* W. Richard Stevens, ISBN 0-201-63346-9, Addison-Wesley Publishing Co., 1994.

[14] *Internetworking with TCP/IP, Volume 1,Fourth Edition,* Douglas E. Comer, ISBN 0-13-018380-6, Prentice-Hall, 1995, 2000.

[15] *HTML 4.0 Sourcebook,* Ian S. Graham, ISBN 0-471-25724-9, John Wiley & Sons, 1998.

[16] *CGI Programming with Perl, Second Edition,* Scott Guelich, Shishir Gundavaram, and Gunther Birznieks, ISBN 1-56592-419-3, O'Reilly and Associates, Inc., 2000.

[17] *JavaScript, The Definitive Guide, 3rd Edition,* David Flanagan, ISBN 1-56592-392-8, O'Reilly and Associates, Inc., 1998.

[18] *Perl Cookbook,* Tom Christiansen and Nathan Torkington, IBBN 1-56592-243-3, O'Reilly and Associates, Inc., 1998.

[19] *Perl for System Administration,* David N. Blank-Edelman, ISBN 1-56592-609-9, O'Reilly and Associates, Inc., 2000.

[20] *HTML 4 for Dummies, Quick Reference,* Deborah S. Ray and Eric J. Ray, ISBN 0-7645-0332-4/1, IDG Books Worldwide, 1998.

Alphabetical Index

/etc/sudoers 29
accessories 15, 16, 23
Apache ... 4, 9, 10, 20, 21, 22, 23, 25, 27, 30, 38, 57
Apache configuration 23
architecture 20
CGI 9, 25, 37
cgi-bin 9
chkconfig 18
cleanup.pl 8
cluster 33
comment_strip 8, 37
compose_IR.cgi 8, 10, 33
condense.pl 8
configuration file 14, 22, 30, 33, 42, 47
crontab 7, 17, 27, 40
custom kernel 13, 21
daily_stats.dat 9
DMZ 26
events of interest 4, 7, 20, 37
fetchem.pl 8, 33, 38, 39
filter 4, 20, 28, 37, 41
find_scan.pl 8, 37
firewall 4, 5, 6, 12, 13, 17, 19, 20, 26, 37, 57
hosts.allow 18, 40
httpd 22, 25, 29, 41
IDE 12
incident report 33
incident reports 20
inner perimeter 6
interface 12, 13, 18, 19, 20
invisible 6, 12, 13, 19
iptables 13, 18, 21, 40, 52
JavaScript 10
kernel loadable modules 14
kill_group.cgi 10
KillWin 11
least privilege 15
libpcap 4
lookup.cgi 10
Menuconfig 22
mod_perl 23
mod_ssl 23
netlog 4, 5
network infrastructure 5
network interface card 12
network services 18, 40
NIC 12, 13
nmap 10, 25, 29, 30, 37, 38
nmap.cgi 10, 29, 38
ntpdate 17, 40
obfuscate.pl 8, 33
openwin.js 10
OpenWindow 11
outer perimeter 6
partition 12, 21
passphrase 27, 28
patch 21
patches 13
public key 15, 16, 26, 27, 28
RAID 6, 20
reconnaissance 29
ReportWin 11
risky stuff 21
RPM 13, 16, 21, 23
SCSI 12
search.cgi 9, 10
SearchWin 11
sensor_driver.pl 7, 17, 18
sensor_init.sh 7, 16, 18
shadow.conf 8, 10, 33, 37
shadow.conf 8
site 33, 34, 36, 37
span 19
SSH 13, 15, 16, 18, 20, 22, 23, 26, 27, 28, 57
ssh_authorized_keys 27
ssh_host_dsa_key 26
ssh_host_key 26
ssh_host_rsa_key 26
ssh_known_hosts 26
start_logger.pl 7
statistical summaries 7, 33
statistics page 9
statistics.ph 33
Stephen Northcutt 4, 57
stop_logger.pl 7
Storable 9
Sudo 30
SUDO 29, 37
tcp wrappers 18
tcpdump 4, 5, 7, 8, 12, 13, 15, 17, 18, 21, 23, 25, 28, 37

time..... 12
tools.cgi..... 9
ToolWin..... 11
viewpoint..... 33
web page 4, 20, 21, 25, 28, 34, 37, 41
whois.cgi.....10
whoiswin.....11
Xconfig 22
xinetd.....18, 40